EESTI MAAÜLIKOOL
Tehnikainstituut

Steven Visla

MUGAVUSMOODUL VOLVOLE

COMFORT MODULE FOR VOLVO

Bakalaureusetöö
Tehnika ja tehnoloogia õppekava

Juhendaja: Janar Kalder, *MSc*

Tartu 2018

LÜHIKOKKUVÕTE

<table>
<tr><td colspan="2">Eesti Maaülikool
Kreutzwaldi 1, Tartu 51014</td><td colspan="2">Bakalaureusetöö lühikokkuvõte</td></tr>
<tr><td colspan="2">Autor: Steven Visla</td><td colspan="2">Õppekava: Tehnika ja tehnoloogia</td></tr>
<tr><td colspan="4">Pealkiri: Mugavusmoodul Volvole</td></tr>
<tr><td>Lehekülgi: 50</td><td>Jooniseid: 23</td><td>Tabeleid: 2</td><td>Lisasid: 2</td></tr>
<tr><td colspan="4">Osakond / Õppetool: Energiakasutuse õppetool
ETIS-e teadusvaldkond ja CERC S-i kood: 4. Loodusteadused ja tehnika, 4.17. Energeetikaalased uuringud, T140 Energeetika;
Juhendaja(d): Janar Kalder, MSc
Kaitsmiskoht ja -aasta: Eesti Maaülikool, 2018</td></tr>
<tr><td colspan="4">Töö eesmärk on luua mugavusmooduli 1999-2014 aasta Volvo mudelitele, mis on ehitatud P2 platvormile. Kuigi moodul on suunatud kindlale platvormile, töötab seade ka uuematel platvormidel väikeste muudatustega. Töös antakse ülevaade mugavusmooduli elektriskeemist ning trükkplaadist. Samuti sõidukimoodulite ühendusest ning mugavusmooduli ühendamisest sõidukiga. Mugavusmooduli eesmärk on lisada sõidukile mugavuslisasid, mida tehases paigaldatud pole, kuid mis on teistel sama-aasta automarkidel juba olemas. Seadme abil kuulatakse ning sisestatakse sõiduki CAN-võrku käsklusi saavutamaks soovitud tulemusi. Töö käigus lisati sõidukile järgnevad lisad: Puldilt akende avamine/sulgemine; uste avamisel/sulgemisel küljepeeglite lahti/kinni klappimine; uste avamisel/sulgemisel lombivalgusti tööle; koos lombivalgustiga põlema ka tagurdustuled; automaatne uste lukustus sõitma hakkamisel. Edasiarenduse võimalusena on välja pakutud arvuti ja seadme vahelise programmi loomist ning USB mikrokontrolleri juhtkoodi muutmist programmiga suhtlemiseks.</td></tr>
<tr><td colspan="4">Märksõnad: CAN-võrk, programmeerimine, mikrokontroller, mugavus</td></tr>
</table>

ABSTRACT

Estonian University of Life Sciences Kreutzwaldi 1, Tartu 51014		Abstract of Bachelor's Thesis	
Author: Steven Visla		Curriculum: Engineering	
Title: Comfort Module for Volvo			
Pages: 50	Figures: 23	Tables: 2	Appendixes: 2
Department / Chair: Chair of Energy Application Engineering Field of research and (CERC S) code: 4. Natural Sciences and Engineering; 4.17. Energetic Research; T140 Energy research. Supervisors: Janar Kalder, *MSc* Place and date: Estonian University of Life Sciences, 2018			
The aim of this thesis is to manufacture comfort module for Volvo P2 platform models. Even though this comfort module is aimed for P2 platform, with slight modifications, it would work on newer platform models. This thesis gives overview of comfort module electrical schematic and circuit board design. Also it gives overview of connections with vehicle's modules and how to connect given comfort module to vehicle. The aim of comfort module is to add functions to vehicle, which is not included from factory but other car manufacturing companies have added to their same year models. Comfort module listens and writes command to vehicle's CAN-bus to add such features. Features added to vehicle: open/close windows from remote; open/close side mirrors when locking/unlocking vehicle; approach lights on when unlocking/locking vehicle; reverse lights on with approach lights; automatic door lock when starting to drive. Further development possibility would be to create computer software to communicate with device to change settings. Also to modify USB microcontroller's program to accept such commands from computer.			
Keywords: CAN-bus, programming, microcontroller, comfort			

SISUKORD

SISSEJUHATUS

Käesoleva bakalaureusetöö eesmärgiks on valmistada mugavusmoodul Volvole. Mugavusmoodul on suunatud vahemikus 1999–2014 väljalastud P2 platvormil ehitatud mudelitele, milleks on S60 (2001–2009), S80 (1999–2006), V70 (2001–2007), XC70 (2001–2007), XC90 (2003–2014) [1]. Kuigi seade on suunatud P2 platvormile, töötab see ka uuematel mudelitel, tehes paar väikest muudatust. Töö läbiviimiseks kasutati 2007. aasta Volvo XC70.

Antud moodul lisab sõidukile mugavuslisasid, mis oleksid võinud olla juba tehase poolt paigaldatud. Paljud sellised lisad on juba teistel, enne 1999. aastat välja antud automarkidel olemas.

Töö esimeses osas kirjeldatakse kasutatud seadmeid, nende otstarvet ning kuidas need aitavad töö valmimisele kaasa. Teine osa sisaldab mugavusmooduli elektriühendusi. Kolmandas osas räägitakse trükkplaadi koostamisest. Töö neljandas osas selgitatakse, kuidas on sõidukiga ühendus loodud ning kommentaare, kuidas seda õigesti teha. Viiendas osas kirjeldatakse mugavusmooduli mikrokontrollerite programmeerimist. Töö kuues osa hõlmab endas andmete töötlust, sealhulgas nende kuulamist, filtreerimist ja saatmist. Seitsmendas osas antakse ülevaade terviklahendusest.

Töö käigus proovitakse sõidukile lisada järgmised võimalused:

- puldilt aknad lahti ja kinni;
- uste avamisel või lukustamisel küljepeeglid lahti või kinni;
- uste avamisel (pimedas) lombivalgusti tööle;
- lombivalgusega koos põlema ka tagurdustuled;
- uste automaatne lukustus sõitma hakkamisel (kuni 2005. a);
- sademete korral automaatne akende sulgemine parkimise hetkel;
- automaatsed istmesoojendused, kui väljas alla 5 °C.

CAN-võrk töötati välja Robert Bosch GmbH (ehk tänapäeval tuntud kui Bosch) firma poolt ja esitleti 1986. aastal Autotööstuste Inseneride ühingu (ingl. *Society of Automotive*

Engineers, tänapäeval tuntav kui SAE) koosolekul [2]. CAN-võrk ühendab kõik moodulid omavahel, et need töötaksid säästlikult ning efektiivselt. Näiteks kaalus sõiduk 45kg vähem, kui kasutati antud võrku ning samuti töötasid andurid märgatavalt kiiremini [2]. Seda sellepärast, et enne võrgu välja töötamist ühendati kõik moodulid eraldi juhtmete abil ning osa infot pidi liikuma läbi mitme mooduli. CAN-võrk eemaldab selle probleemi, kuna kõik moodulid on nii-öelda ühele tasandile ühendatud ehk ükski moodul ei pea suhtlema läbi teise mooduli.

Kõik moodulid saavad lugeda ja kirjutada võrku. Peamised eelised sellise võrgu puhul on madal hind (ei ole vaja teha mitut võrku, et kõik moodulid suhelda saaksid), tsentraliseeritus (ühe võrgu abil on võimalik kõikidele moodulitele ligi pääseda), töökindlus (näiteks ei ole tundlik magnetväljadele), efektiivsus (võrgus liikuvad paketid on prioriseeritud ehk tähtsamad andmed saavad kõigepealt edastatud), paindlikkus (igal seadme oma võrgust lugemise ja võrku kirjutamise võimalus, seega iga moodulit on võimalik teha täpselt selliselt nagu soovitakse) [3]. Andmete edastuseks kasutatakse kahte juhet, mis on omavahel keerdus (nimetatakse keerdpaariks) ning mille pikkused võivad olla kuni 1000 meetrini, sellisel juhul võimalik kasutada kiirust 40 kbps, kuid näiteks 40 meetril võimalik kasutada kiirust 1 Mbps [4].

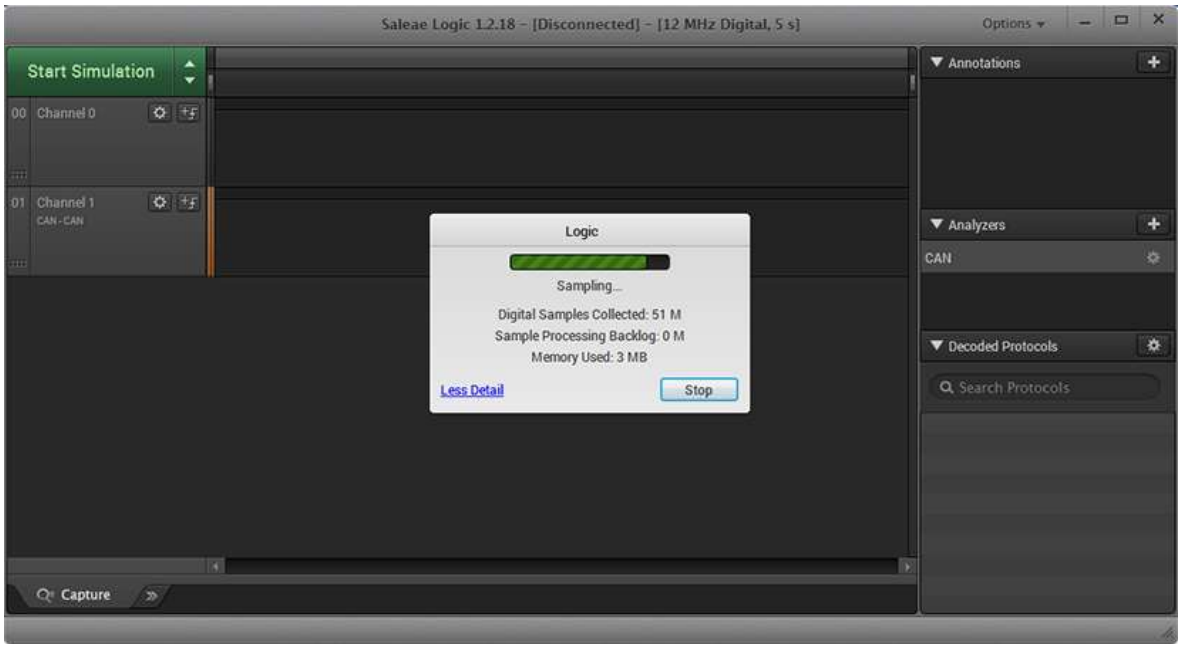
CAN-võrgu andmejuhtmed peavad olema omavahel keerdus ehk nii-öelda võrgu andmejuhtmed on keerdpaaris. Vastasel juhul võrgu signaal kaob seal juhtmes ära, ning võrgust ei ole võimalik lugeda ega sinna kirjutada [5].

1. Seadmete kirjeldus

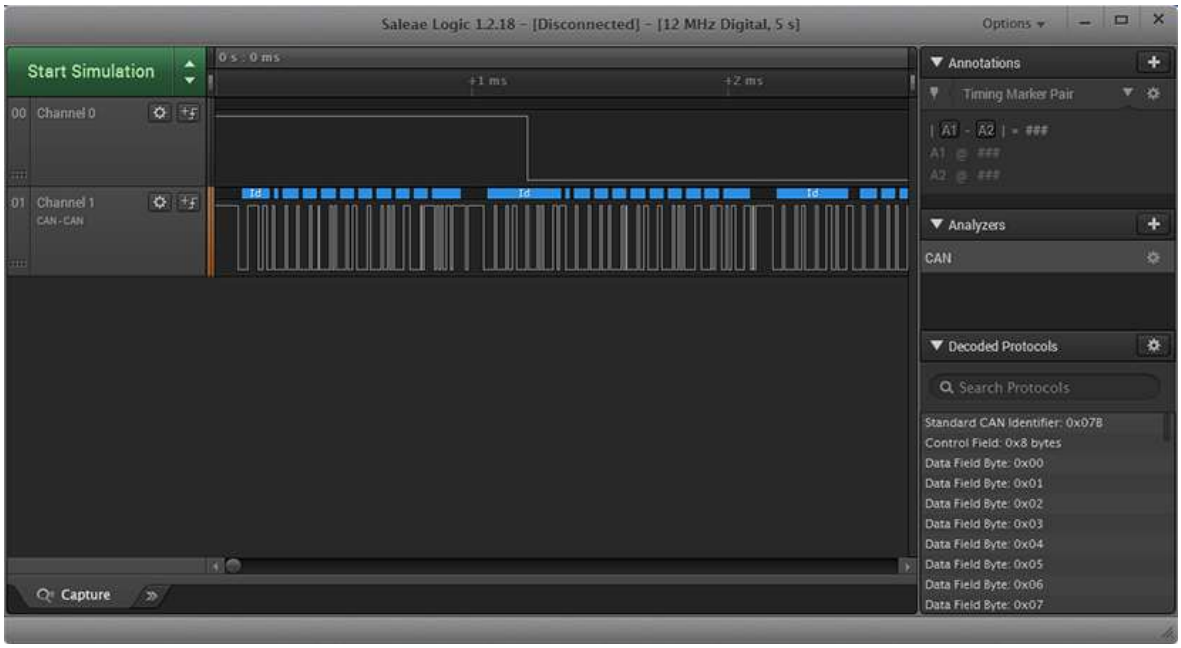
Käesolevas töös on kõige enam kasutavateks seadmeteks Saleae 8-kanaliline loogika analüsaatori kloon, mida kasutatakse igasuguste andmete, loogika, protokollide ning sageduste analüüsimiseks, ning multimeeter HoldPeak HP-33D. Lisaks on kasutusel veel Volvo VIDA diagnostikatarkvara. Selles projektis on multimeetrit HoldPeak vaja pingete ning ühenduste kontrollimiseks, et olla kindel kõikide ühenduste korrektses ühendusviisis. Käesolevas projektis on loogika analüsaatori abil jagu saadud pikaaegsest kommunikatsiooni probleemist, mis tulenes valel infol, millest räägitakse täpsemalt esimese peatüki lõpus. Loogika analüsaator on kujutatud joonisel 1.1.

Joonis 1.1. Saleae loogika analüsaator.

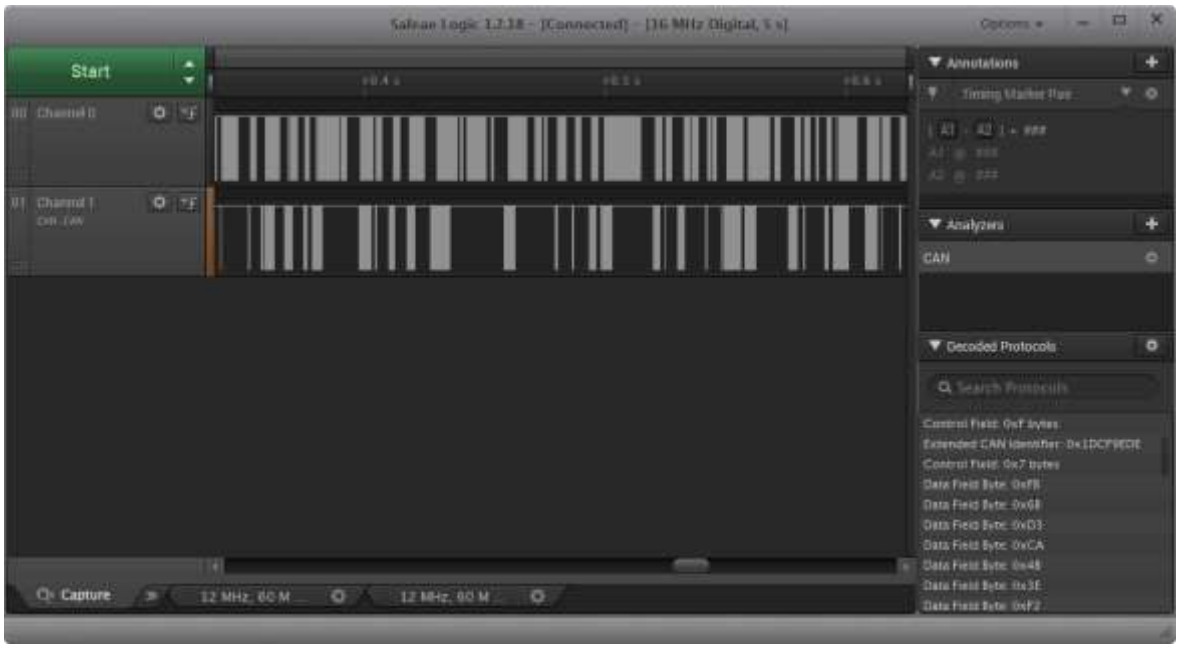
Kuigi kasutatud loogika analüsaator on kloon, töötab see originaaltarkvaraga. Tarkvara on tasuta kättesaadav Saleae kodulehelt. Programmis on võimalik kasutada demorežiimi, mis on mõeldud programmiga tutvumiseks. Näidet, kuidas demorežiimis andmeid kogutakse, on kujutatud joonisel 1.2 ning demo andmeid on näha joonisel 1.3. Järgmisel joonisel, joonis 1.4, on kujutatud reaalsest sõiduki kommunikatsioonist ühte osa, et anda põgus ülevaade, milline näeb välja edastatud andmepakett.



Joonis 1.2. Saleae tarkvara Demo režiimi andmete kogumine.



Joonis 1.3. Saleae tarkvara Demo reziimi andmete kuvamine.



Joonis 1.4. Saleae tarkvara poolt kogutud üks osa sõiduki CAN-võrgu suhtlusest.

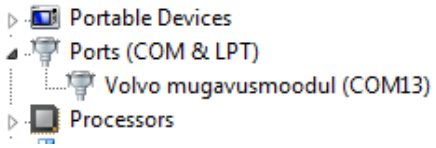
Loogika analüsaatori abil on töö autor kindlaks teinud, et kuigi eelnevalt kogutud informatsiooni põhjal oldi kindel, et mainitud sõiduki suhtluse kiiruseks on 250 kbps, siis tegelikult on selleks siiski 125 kbps.

Volvo diagnostikatarkvara VIDA kasutatakse selleks, et teada saada, kuhu on vaja teha vajalikud ühendused, et mugavusmoodul saaks endale toite ning saaks sealjuures suhelda sõiduki CAN-võrguga.

Mugavusmooduli mikrokontrollerite programmeerimiseks kasutatakse programmeerijat USBasp ning tarkvara AVRdude. USBasp on otse USB pessa ühendatav seade, mille abil on võimalik programmeerida kõiki Atmel AVR mikrokontrollereid, mis aktsepteerivad üle SPI (ingl. *Serial Peripheral Interface*) liidese programmeerimist [6]. Ainukeseks piirajaks on tarkvara. Tarkvara AVRdude on konsoolipõhine programm, mis võimaldab kirjutada mikrokontrolleritele peale uut juhtkoodi. USBasp on kujutatud joonisel 1.5.

Joonis 1.5. USBasp programmeerija.

Kuigi trükkplaadil on ka USB kommunikatsiooni mikrokontroller, siis antud töö käigus seda ei kasutata. Kasutamiseks tuleb luua vastav arvutipoolne tarkvara, mis oskaks suhelda antud seadmega. Samuti tuleb USB ja loogika mikrokontrolleri juhtkood modifitseerida selliselt, et see muudaks soovitud parameetreid vastavalt arvuti poolt saadetud konfiguratsioonile. Ajutiselt on USB mikrokontrollerile peale kirjutatud juhtkood, mis ainult tutvustab arvutile ennast kui „Volvo mugavusmoodul“. Ajutine juhtkood on modifitseeritud LUFA projekti (vana nimega MyUSB) kood, mis on välja antud Dean Camera poolt [7]. LUFA projekt sisaldab endas tarkvara, dokumentatsiooni ning näiteid, et muuta ATmega16U2 kivi USB seadmeks, mida arvuti tunnistab [7]. USB mikrokontrolleri juhtkood on modifitseeritud kasutades LUFA CDC näidet. CDC tähendab *Communication Device Class*, mille abil on võimalik tekitada mikrokontrollerist virtuaalne jadaport (*COM-port*), mis omakord võimaldab arvuti ning seadme vahel tekitada suhtlust [8]. Loogika poolel olevale mikrokontrollerile on programmeeritud autori enda koostatud kood, millest räägitakse täpsemalt peatükis 5. Joonisel 1.6 on näha, et arvuti tunneb seadet, kui „Volvo mugavusmoodul“.



Joonis 1.6. Seade tuntakse arvuti poolt ära õige nimetusega.

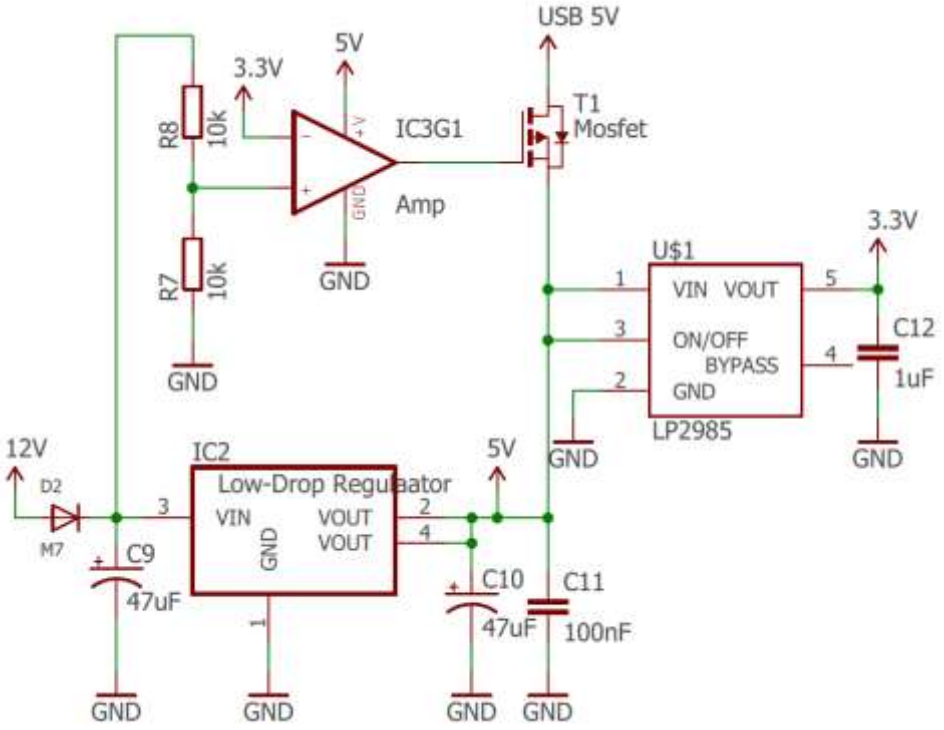
2. Mugavusmooduli elektriskeem

2.1. Elektriskeemist üldiselt

Elektriskeemi koostamisel on lähtutud mikrokontrollerite ja -kivide andmelehtedel olevast informatsioonist ning praktilisest kogemustest. Elektriskeem on koostatud programmiga Autodesk Eagle, mis on tudengitele õppeperioodiks tasuta kasutatav. Elektriskeem on jagatud kolmeks: toite reguleerimine, loogika ja kommunikatsioon. Elektriskeemi üldvaade on toodud lisas A.

2.2. Toite reguleerimine

Toite reguleerimise osas tekkinud selektiivsus sõiduki ning USB kaabli toite vahel on lahendatud selliselt, et kui sõiduki poolt toidet pole, siis saab seade voolu USB kaablist. Kui mingil hetkel saab seade signaali, et sõiduki poolt on samuti toide olemas, lülitutakse automaatselt ümber sõidukitoitele. Samuti on lisatud pingeregulaator, mis teeb sõiduki 12 volti (tegelikult umbes 14 volti) mikrokontrolleritele ja -kividele sobivaks viieks voldiks. Toite reguleerimise ülesehitusest annab ülevaate joonis 2.1.



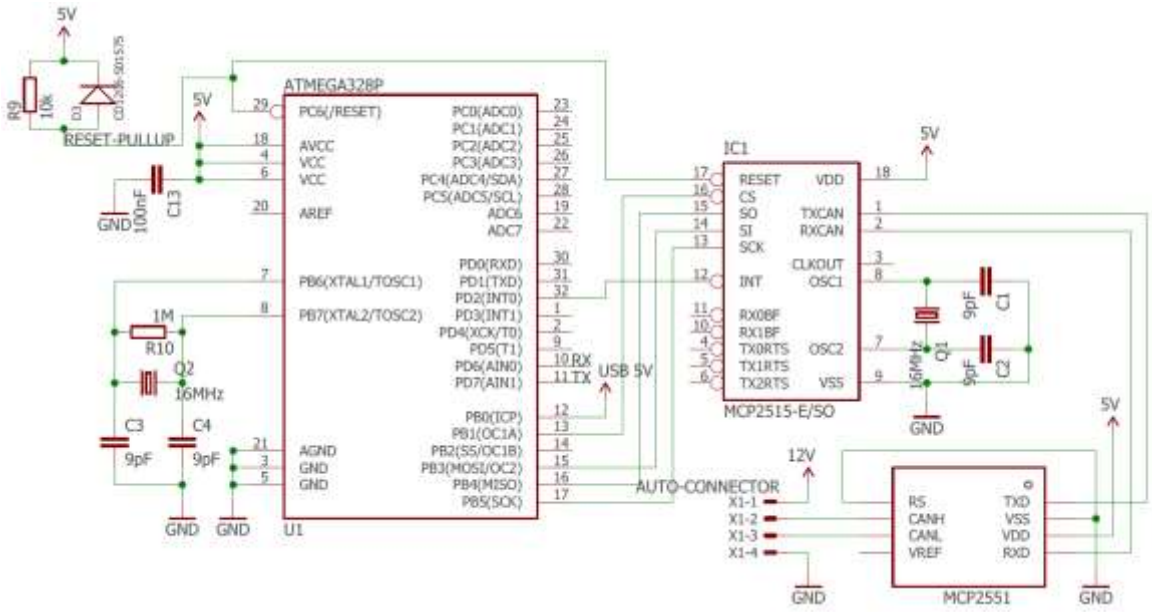
Joonis 2.1. Mugavusmooduli elektriskeemi toite reguleerimise osa.

Joonisel 2.1 tähistab 12V (vasakul) sõiduki toite sisendit. Paremal genereeritakse LP2985 abil 3.3V ainult operatsioonivõimendi (joonisel IC3G1) jaoks kasutatavaks pingete vaheks. Sisuliselt on tööpõhimõtteks see, et kui sõiduki poolt tuleb toide, siis operatsioonivõimendi edastab *MOSFET*'ile (joonisel T1) positiivse pinge mille tõttu lülitatakse *MOSFET*'i, mis otsustab kas kasutatakse USB toidet või mitte. Antud lahendus on kasulik juhul, kui soovitakse sisse viia muudatused seadme käitumises ning kui seade pole ühendatud sõiduki toitega ja muudatuste kontrollimiseks on vajalik ühendada seade sõidukiga samal ajal, kui USB juhe on ka ühendatud.

2.3. Loogika

Loogika poole peal asuvad sõidukiga suhtlemiseks vajalikud mikrokivid ning suhtlust juhtiv mikrokontroller. Mikrokontrollerina on kasutatud ATmega328P põhjusel, et see on kõige lihtsamini kättesaadav. Loogika osal on ka *reset* liinil *pullup* takisti, mida kasutab nii loogika kui ka kommunikatsiooni pool. *Pull-up* takisti eesmärk on hoida liini, kuhu takisti on

ühendatud, pinge all selle hetkeni, kuni rakendatakse liinile maandus ehk 0 volti. Loogika osa elektriskeemi on näha joonisel 2.2.

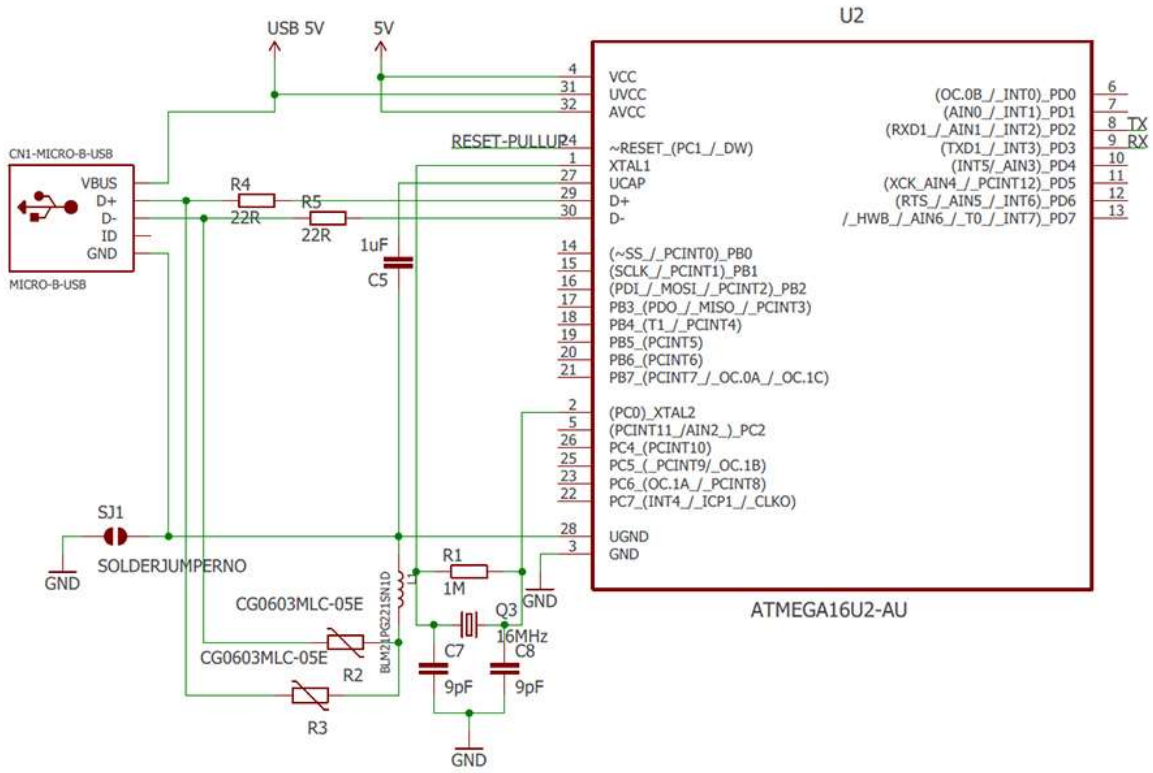


Joonis 2.2. Mugavusmooduli elektriskeemi sõiduki loogika osa.

Joonisel 2.2. olev MCP2551 mikrokivi vastutab sõiduki CAN-võrgu kuulamise eest. MCP2551 kivi edastab saadud andmed üle RX ja TX siini MCP2515 mikrokivile. Kui MCP2515 kivi saab andmed kätte, filtreerib mikrokivi andmeid ning sobiva paketi ID-koodi puhul annab sellest teada mikrokontrollerile, muutes väljundi INT madalaks (ehk väljundil on 0 volti), mida mikrokontroller pidevalt kontrollib.

2.4. Kommunikatsioon

Kommunikatsioon hõlmab endast suhtlust arvuti ja mikrokontrolleri vahel. Arvutiga suhtlemiseks on kasutatud ATmega16U2 mikrokontrollerit. Antud mikrokontrolleri jagab loogika mikrokontrolleriga sama *reset* ühendust. Selle mikrokontrolleri abil saab seadistada seadme käitumist vastavalt oma soovidele. Näiteks kas uste lukustamisel klapitakse ka küljepeeglid, kas sõidukit lukust lahti tehes pimedal ajal pannakse tuled automaatselt põlema (ja kas kõik võimalikud või vaid valitud tuled) või kas on võimalik puldist aknaid avada. Kommunikatsiooni elektriskeemi on näha joonisel 2.3.



Joonis 2.3. Mugavusmooduli elektriskeemi arvutiga suhtluse osa.

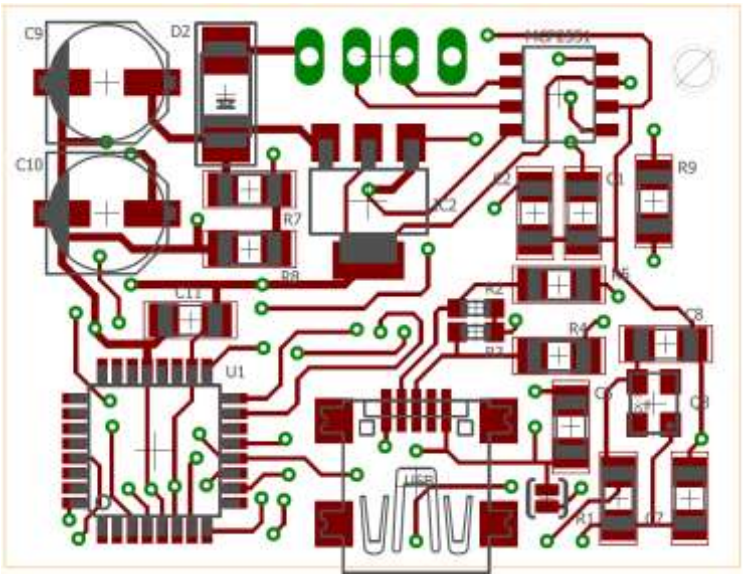
Joonisel 2.3 olevad takistid R4 ja R5 on USB andmeliinide lõpptakistid, mis tagavad, et ei tekiks andmete tagasipeegeldumist. Joonisel olevad varistorid (R2 ja R3) pole tegelikult vajalikud, sellest on täpsemalt räägitud järgmises peatükis.

3. Mugavusmooduli trükkplaat

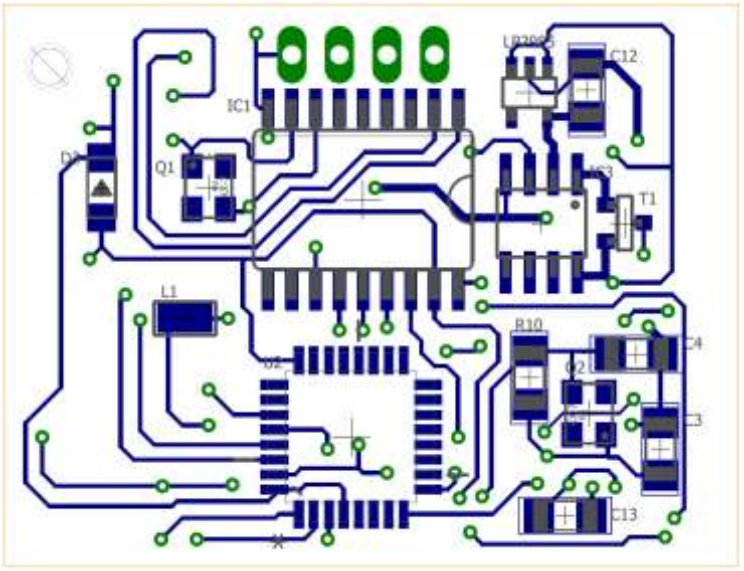
Mugavusmooduli eduka toimimise jaoks ei piisa vaid elektriskeemist, on vaja, et seade oleks olemas ja töötaks ka füüsiliselt. Selle jaoks kujundati eraldi trükkplaat. Trükkplaadi kujundamisel lähtuti mikrokontrollerite ja -kivide andmelehtedest. Trükkplaadi kujundus koostati samuti programmiga Autodesk Eagle nagu elektriskeemi puhul. Trükkplaadi valmistamine telliti firmalt SeeedStudio, mis paikneb Hiinas. Komponendid telliti läbi portaalide eBay, AliExpress ning Farnell. Komponendid paigaldas trükkplaadile töö autor kasutades jootekolbi ning kuumaõhujaama.

Kokku on trükkplaate koostatud neli erinevat versiooni erinevate kujundustega, et saavutada väikseim võimalik suurus. Lõppversiooni trükkplaadi kogumõõtmeteks on 40x30 mm ning sealjuures on plaat kahepoolne. Ilmselt oleks olnud võimalik teha mõõtmete poolest veel väiksem lahendus, kui oleks kasutanud nelja- või enamakihilist trükkplaati, kuid sellisel juhul oleks hind muutunud kallimaks. Kahekihilisel, kuni mõõtmeteni 100x100 mm, on kümne trükkplaadi hind 4.90€, kuid neljakihilisel, samade parameetritega, oleks kümne plaadi hind 49.90€ ehk üle kümne korra kallim [9]. Mõõtmetel 20x20mm jääb kahekihilise hind samaks (võrreldes 100x100mm suurusega), kuid neljakihilisel langeb 39.90 euroni, kuid ikkagi on see üle kaheksa korra kallim [9]. Hindade arvutusel on kasutatud firma kodulehel leitavat hinnakalkulaatorit. Miinimum kogus trükkplaatide tellimisel oli 10 tükki.

Trükkplaadi kujundused on kujutatud joonisel 3.1 ja joonisel 3.2, kus joonis 3.1 on plaadi pealtvaade ning joonis 3.2 on plaadi altvaade.



Joonis 3.1. Trükkplaadi kujunduse pealtvaade.

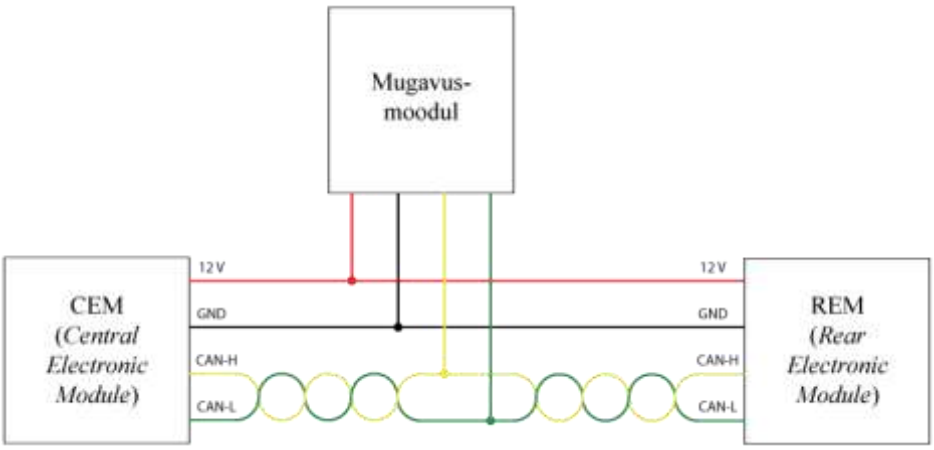


Joonis 3.2. Trükkplaadi kujunduse altvaade.

Peale komponentide peale jootmist ja trükkplaadi arvutiga ühendamist ilmnes asjaolu, et joonisel 3.1 olevad kaks varistori (R2 ja R3) segavad arvuti ja seadme vahelist suhtlust. Ilma nendeta suhtlus töötab. Sellega seoses ei ole vaja ka joonisel 3.2 olevat pooli L1, kuna see on ühendatud vaid varistoride ning maanduse vahele.

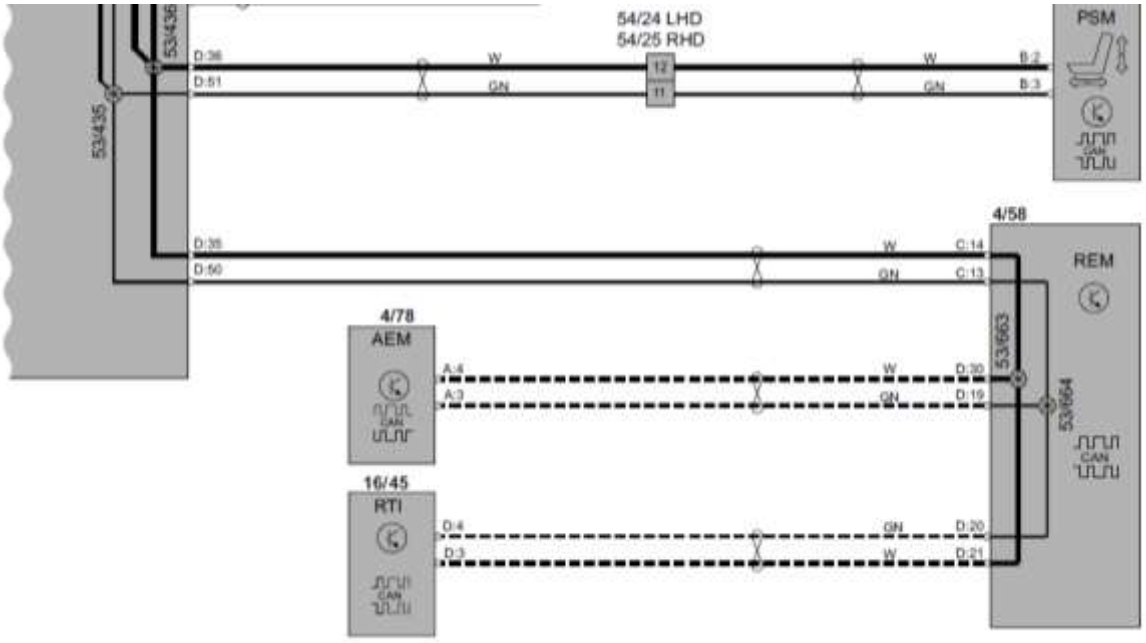
4. Ühendus autoga

Autoga on ühendatud kokku neli juhet, millest kaks on toitejuhtmed ja kaks on CAN-võrgu juhtmed. Kõik ühendused on tehtud olemasolevatele juhtmetele ehk on lisatud rööpselt juurde. Meeles tuleb pidada, et CAN-võrgu juhtmed peavad olema keerdpaaris, et suhtlus töötaks. Toitejuhtmete valimisel valiti selline 12 voldi juhe, kus on vool pidevalt sees. Seda sellepärast, et seade kasutab puhkerežiimis vähe voolu ning sellisel juhul on seade kohe olemas, kui seda vajatakse. Ühenduse illustreerimiseks on joonis 4.1.



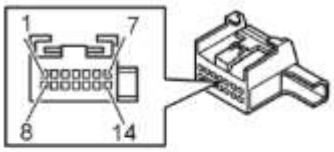
Joonis 4.1. Sõiduki ning mugavusmooduli ühenduse põhimõtteskeem.

Õigete juhtmete leidmiseks kasutati Volvo diagnostika tarkvara, kus oli võimalus tutvuda sõiduki juhtmestiku ja moodulitega. Ühendus tehti REM (*Rear Electronic Module*) juures kerge ligipääsu tõttu. Diagnostika tarkvaras oli võimalik eraldi CAN-võrgu liinid kuvada ning näha, millisesse mooduli sisendi alla võrk ühendub (REM mooduli puhul on selleks sisendid 13 ja 14) [10]. CAN-võrgu liinid meid huvitava mooduli juures on kuvatud joonisel 4.2.



Joonis 4.2. CAN-võrgu ühendused sõiduki mooduli juures, kuhu ühendatakse mugavusmoodul [10].

Kuigi võib eeldada, et joonisel tumedamalt toodud liin on nö CAN *High* (Kõrge), tuli see igaksjuhuks üle kontrollida. Kontrollimiseks on vaja diagnostika tarkvaras määrata sõiduki profiil ning lahti võtta mooduli ühenduste tabel. Leides sobiv mooduli, leiti kinnitus, et liin 14 on CAN-võrgu *High* liin ja liin 13 on *Low* liin [10]. Samuti on kujutatud seal, kus antud liinid pistiku küljes asetsevad. Antud tabelit kirjeldab joonis 4.3.



Connector C					
Breakout box terminal	**Control module terminal**	**Signal type**	**Ignition off** Active / Not active	**Ignition on** Active / Not active	**Miscellaneous**
#41	#C1	Alarm LED	U = approximately 1.6V/ U_{low}	U_{low}	Pulsed signal, f = approximately 0.5Hz
#42	#C2	Reserve			
#43	#C3	Switch left-hand rear door, signal	U_{low}	U_{low}	Grounded with door closed, open-circuit with door open
#44	#C4	Hood switch	U_{low}	U_{low}	Grounded with door closed, open-circuit with door open
#45	#C5	Lock switch left-hand rear door, signal	U_{bat} if door unlocked; U_{low} if door locked	U_{bat} if door unlocked; U_{low} if door locked	
#46	#C6	Gas discharge lamp module (GDL) sensor			
#47	#C7	LIN communication	During communication the voltage is 0 V - U_{bat}	During communication the voltage is 0 V - U_{bat}	
#48	#C8	Glass breakage loop, pulsed signal	0-12 V	U_{low}	
#49	#C9	Signal ground, position sensor for headlamp beam height control	U_{low}	U_{low}	
#50	#C10	Lock switch child proof lock left-hand rear door, signal	U_{low} / U_{bat}	U_{low} / U_{bat}	
#51	#C11	Power supply, position sensor for headlamp beam height control	U = 5 V	U = 5 V	
#52	#C12	Switch left-hand front door, signal	U_{low}	U_{low}	Grounded with door closed, open-circuit with door open
#53	#C13	Communication CAN L low speed network	During communication the voltage is 1.5 - 2.5 V	During communication the voltage is 1.5 - 2.5 V	
#54	#C14	Communication CAN_H low speed network	During communication the voltage is 2.0 - 3.5 V	During communication the voltage is 2.0 - 3.5 V	

Joonis 4.3. VIDA tarkvaras REM ühendusi ühte osa iseloomustav tabel [10].

Seadet on võimalik ühendada ka sõiduki OBD-II (*On-Board Diagnostics*) pistikusse, kuid selline tegevus hõivaks selle pistiku ning iga nelja sekundi tagant peaks saatma võrku nii-öelda „hoia ärkvel“ (tõlge *keep-alive*) sõnumi, vastasel juhul sulgeb sõiduk OBD-II pistikusse minevad andmeühendused. OBD-II pistiku abil on võimalik jälgida ja muuta sõiduki aju ning muid seadmeid, mis on sinna ühendatud. Samuti kasutatakse seda sõidukil esinevate vigade leidmisel [11].

Ühenduste tegemiseks on kasutatud ühendusklambrit, millega on võimalik olemasolevale juhtmele lisajuhe juurde lisada, ilma et peaks juhet katki lõikama või seda muud moodi kahjustama. Ühendusjuhtmetele on lisatud ka pistik ning pesa juhuks, kui soovitakse seade ajutiselt sõidukist eemaldada. Klambrit iseloomustab joonis 4.4.

Joonis 4.4. Olemasolevale juhtmele lisajuhtme juurde lisamiseks kasutatav klamber.

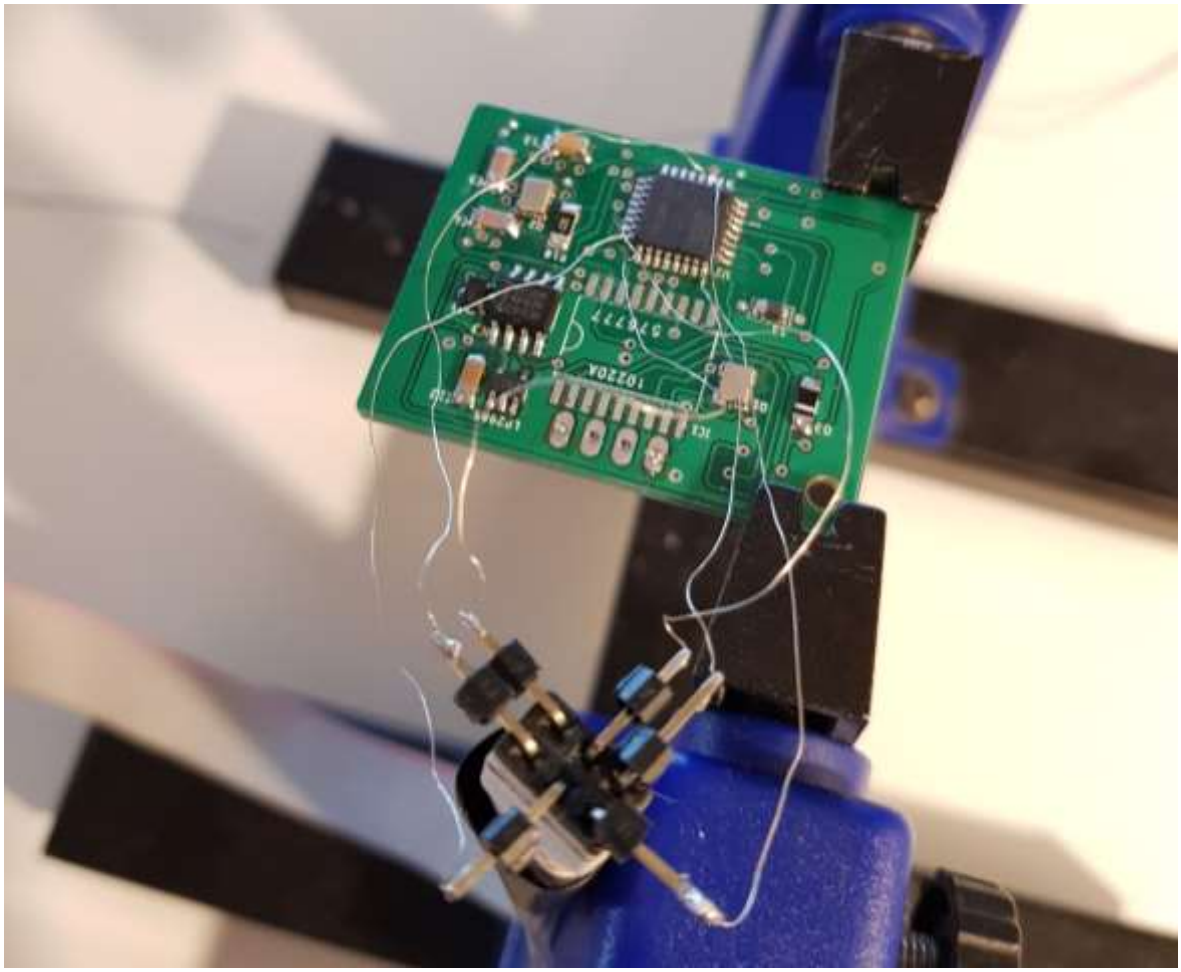
Peale ühenduste tegemist kontrolliti iga lisatud juhe eraldi üle: oluline oli uurida, kas juhtmed annavad korrektselt ühendust ning pinged on sellised, nagu olema peavad. Kontrollimiseks kasutati multimeetrit. Lähtudes joonisest 4.3, teame ka, et CAN-võrgu pinged peavad olema 2.0-3.5 voldi vahel (kui võrk on aktiivne). Seda tuleb mõõta maanduse ja CAN-võrgu liini vahelt, nii CAN madalal kui kõrgel eraldi. Toitepinge, kui sõiduk ei käi, peab olema sama suur kui akupinge ning kui sõiduk käib, siis peab olema laadimispinge, mis on keskmiselt 14 volti. Sõidukipoolset ühendust iseloomustab joonis 4.5. Joonisel keerdpaaris olev roheline ning valge juhe on CAN-võrgu omad ning eraldi olev punane juhe on pideva 12 voldi jaoks ning valge on maanduse jaoks.

Joonis 4.5. Sõidukipoolne toite ning CAN-võrgu ühendus.

Jooniselt on näha, et näha, et seadmesse ühenduv CAN-võrgu juhtmed on endiselt keerdpaaris ning näha on ka, kuidas on veetud toitejuhtmed.

5. Programmeerimine

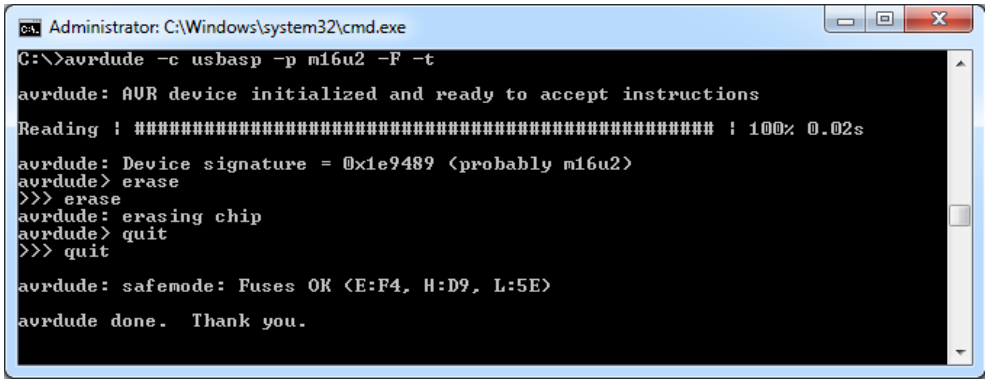
Telliti täiesti tühjad mikrokontrollerid, mis vajasid veel programmeerimist. Programmeerimiseks kasutati USBasp programmeerijat, AVRdude tarkvara ning juhtkoode. Mikrokontrolleri ühenduseks kasutati juhtmekiude, mis ühendati vajalike jalgadega ning USBasp-ga. Ühendust mikrokontrolleriga iseloomustab joonis 5.1. Joonisel on kujutatud ATmega16U2 esmane programmeerimine.



Joonis 5.1. Mikrokontrolleri programmeerimiseks tehtud ühendused mikrokontrolleri ning programmeerija USBasp vahel.

Joonisel 5.1 all nurgas on USBasp juhtmeots, kuhu on võimalik ühendada programmeerimiseks vajalikult mikrokontrolleri ühendused. USBasp juhtmeots on kinnitatud teibiga trükkplaadi hoidiku külge, et juhtmekiud ei liiguks ja ei tekitaks lühist. Lühis võib tekkida hetkel, kui USBasp-d ühendadakse USB pessa või eemaldatakse seda sealt. Sama tehnoloogiat kasutades teostati ka teisel pool plaati olevale mikrokontrollerile esmane programmeerimine.

Programmeerimiseks on kasutatud programmeerijat USBasp ning tarkvara AVRdude. Kuna tegemist on täiesti tühja ja varem programmeerimata mikrokontrolleritega, siis kõigepealt tuleb eemaldada neilt programmeerimise lukustus. Töös tehti seda kasutades avrdude käsklust „-t“, mis sisenes terminal režiimi. Näitena kasutame ATmega16U2 mikrokontrollerit. Seega tuli konsooli sisestada „*avrdude -c usbasp -p m16u2 -t -F*“. Valikut „-F“ kasutasime eeldusel, et kui mikrokontrolleri signatuur ei vasta sellele, mis AVRdude ootab, siis ei jäetaks tööd katki. Seejärel sisestati käsklus „erase“, mis kustutab mikrokontrolleril kõik andmed. Peale seda sisestati käsklus „quit“, mis väljus AVRdude terminali režiimist. Ülevaate antud funktsioonist annab joonis 5.2.



Joonis 5.2. Mikrokontrollerilt programmeerimise lukustuse eemaldamine.

Järgmiseks seadistati mikrokontrolleri fuse-bitid (ingl *fuse bits*). Nendega on võimalik seadistada mikrokontrollerit vastavalt konfiguratsioonile. Näiteks on võimalik panna mikrokontroller käima sisemise kvartskristalli pealt või siis seadistada mikrokontroller välisele kristallile, et saada suurem kiirus.

Fuse-bitide seadistamiseks kasutasime AVRdude käsklust „-U“ mis tähendab, et mälu andmeid hakatakse muutma. Antud käskluse süntaks on „*memtype:op:filename*“, kus

memtype näitab, millist mäluosa hakatakse muutma, *op* näitab operatsiooni liiki (näiteks w on *write* ehk kirjutamine, r on *read* ehk lugemine) ning *filename* tähistab faili, mida hakatakse kirjutama või faili nime kuhu andmed loetakse [12]. Töös on kasutatud *filename* asemel lihtsalt HEX formaadis numbrit.

Mikrokontrollerile taheti seadistada järgmised fuse-bitid: lfuse 0xFF, hfuse 0xD9 ja efuse 0xF4.

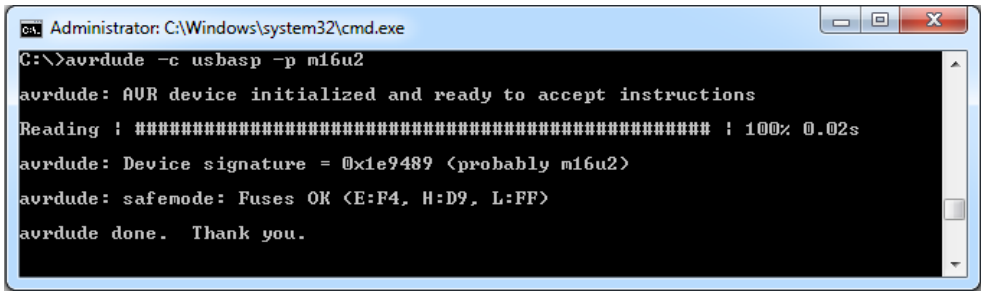
Selline konfiguratsioon seadistab mikrokontrolleri [13]:

- välisele kristallile;
- kristalli signaali ei väljastata PORTC7 väljundisse;
- kristalli väärtust ei jagata kontrolleri siseselt kaheksaga;
- olekukontrolli (*Watchdog timer*) pole aktiveeritud;
- lubatakse üle SPI liidese programmeerimine;
- *reset* sisend on aktiivne;
- miinimum töötamisepinge on 3.0v;
- riistvaraline käivitus on lubatud.

Seega edastatav käsklus oli „*avrdude -c usbasp -p m16u2 -U lfuse:w:0xff:m -U hfuse:w:0xd9:m -U efuse:w:0xf4:m*“. Laiend „:m“ näitab, et väärtus on *filetype* asemel kirjutatud ning seetõttu ei pea looma eraldi faili. Ülevaate antud funktsioonist annab joonis 5.3.

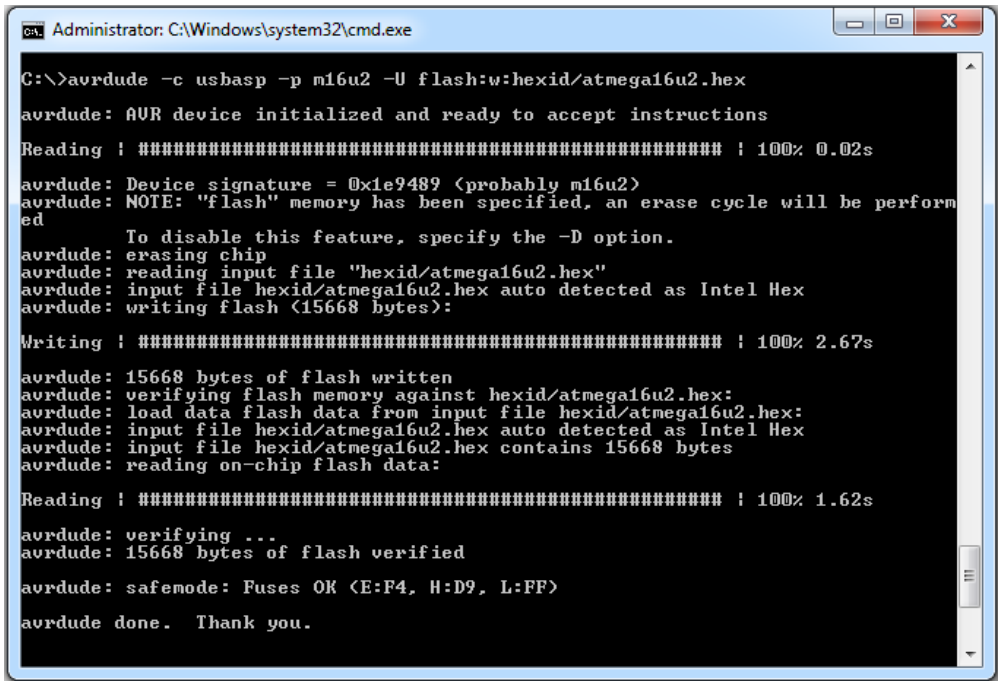
Joonis 5.3. Mikrokontrolleri fuse-bitide muutmine.

Peale fuse-bitide muutmist kontrolliti üle, kas need on tõepoolest muutunud. Kontrollimiseks sisestati lihtsalt programmeerija tüüp ning programmeeritav mikrokontrolleri tüüp. Ehk kasutati käsklust „*avrdude -c usbasp -p m16u2*“. Käsklus andis teada, et mikrokontroller on ühendatud ning seejuures tagastati ka mikrokontrolleri signatuur. Samuti väljastatakse fuse-bitid, mis on tõepoolest muutunud võrreldes joonisega 5.2. Ülevaate antud funktsioonist annab joonis 5.4.



Joonis 5.4. Mikrokontrolleri fuse-bitide muutuse kontrollimine.

Järgmiseks kirjutati mikrokontrollerile juhtkood. Juhtkood kompileeriti Atmel Studiot kasutades valikut „*Build Solution*“, mis kompileeris terve lahti oleva projekti kasutades varem seadistatud sätteid. Juhtkoodi peale kirjutamine käib kasutades AVRdude „-U“ käsklust. Sel korral on *memtype* tüübiks määratud valik *flash* ehk andmed kirjutatakse mikrokontrolleri koodimällu. Samuti on sel korral *filename* asemel kirjutatud tee peale kirjutava juhtkoodi juurde. Seega on edastatav käsklus „*avrdude -c usbasp -p m16u2 -U flash:w:hexid/atmega16u2.hex*“. Ülevaate antud funktsioonist annab joonis 5.5.



Joonis 5.5. Mikrokontrolleri juhtkoodi peale kirjutamine.

Jooniselt on näha, et juhtkoodi programmeerimine õnnestus. Samuti on fuse-bitid õigeks jäänud. Analoogset meetodit kasutades programmeeriti ka teine mikrokontroller.

6. Andmete töötlus

6.1. Andmete kuulamine ja filtreerimine

CAN-võrgu pakett sisaldab endas nii-öelda identifitseerimis ID-d, mis on igal moodulil erinev ja sealjuures unikaalne. Vastasel juhul ei saaks vastava mooduli poole eraldi pöörduda. Näiteks, kui kõik uksemoodulid kannaksid sama ID-d, siis juhul, kui soovitakse kõrvalistuja akent lahti teha, avanevad kõik aknad. Seda sellepärast, et kõik moodulid arvavad, et see pakett on mõeldud neile.

Peale unikaalse ID on paketis veel ka andme ala (8 byte-i), mis sisaldab saadetavat informatsiooni, mida antud moodul edastada soovib. Kuigi pakett sisaldab endas rohkemgi, siis käesolevas töös vaadatakse vaid neid kahte, kuna soovitud tulemuseks rohkem vaja ei ole.

Kõigepealt jälgiti ja koguti kogu andmeliiklust moodulite vahel. Kuna erisuguseid andmeid tuli keskmiselt iga viie millisekundi järelt (0.005 sekundit) ehk sekundis edastati andmeid 200 korda, siis mõisteti kiirelt, et antud monitooringuviisi kasutades pole võimalik leida soovitud mooduli ID koodi. Selle lahendamiseks kirjutati uus kood, mis jätab meelde kõik võrgus esinenud moodulite ID koodid ning millega oli võimalik teha valik, kus kuvatakse vaid kindla mooduli poolt edastatud andmeid.

Jälgides vaid kindlat moodulit, on võimalik tuvastada millisele moodulile antud ID kood kuulub. Tuvastamiseks kasutati lihtsat moodust, kus vajutati vastavat nuppu, kuniks muutus oli tuvastatav. Näiteks klõpsutati roolil raadiokanali edasi vahetamise nuppu, kuni leiti, et moodulis, millel on ID kood 0x0131726C, muutus kaheksas bait 0x3F-ilt 0x3D-ks. Kui nupp lahti lasti, muutus bait esialgseks.

Sarnast metoodikat kasutades leiti üles ka teised vajalikud moodulid. Ainukeseks tingimuseks moodulite leidmisel on asjaolu, et moodul peab olema nö töörežiimis ehk sisuliselt peab olema süütelukk vähemalt esimeses asendis andmete kogumiseks.

Mooduleid, millel polnud võimalik visuaalselt vahet teha, kuna väärtused muutusid nii kiirelt, otsiti viisil, et logiti kogutud andmed faili ning seejärel teostati nii-öelda järel töötlus andmestikule sarnaselt ülaltoodule. Ainukeseks vaheks on see, et andmeid saab ühe rea kaupa üle vaadata. Antud meetod on väga ajakulukas. Kuna kõiki mooduleid, mida töö sooritamiseks vaja oli, ei olnud võimalik üles leida, võeti appi diagnostika tarkvara. Diagnostika tarkvaras on võimalik pärida vastavatelt moodulitelt infot, mis edastatakse vaid päringu peale. Kuulates pealt diagnostika ning sõiduki suhtlust, leiti osad puudu olevad moodulid ning leiti ka, millise käskluse (ehk paketti) saatma peab, et sellist informatsiooni saada.

Kokku kasutati info ja andmete kogumiseks kolme eri metoodikat. Esimeseks on sõiduki CAN-võrgu pealt kuulamine, et märgata samalt aadressilt saadetud pakettides erinevust. Teiseks on CAN-võrgu suhtluse logimine faili ning hiljem selle töötlemine, mis on väga ajakulukas. Kolmandaks on diagnostika tarkvara ning sõiduki vahelise suhtluse pealt kuulamine.

6.2. Andmete saatmine

Osade andmete ja käskluste saatmise jaoks tuleb teada, mida saatis teatud moodul kõige viimasena, kuna osadel moodulitel edastatakse ka nii-öelda *rolling counter* ehk iga andme saatmise ajal teatud bait suureneb. Seetõttu tuleb pidevalt jälgida, mida mingi moodul viimati saatnud on ja edastatud andmed meelde jätta. Näiteks kliimamoodul, mille ID on 0x217FFC, muudab vaheldumisi esimest baiti 0x03 ja 0x01 vahel. Seda (*rolling counter*'it) tehakse sellel eesmärgil, et teised moodulid, mis kuulavad antud moodulit, saaksid aru, kas edastatud on uusi andmeid võrku.

Sellise informatsiooniga saab hakata koostama paketti, mida seade võrku soovib saata. Paketi saatmiseks peab teadma, mis andmeid tuleb saata soovitud tulemuse jaoks. Näitena

kasutame endiselt kliimaplokki, mille ID on 0x217FFC. Kliimaploki poolt edastatud andmed tavaolekus on toodud tabelis 6.1 ning sõiduki küljepeeglite klappimisnupu vajutusel edastatud andmed on toodud tabelis 6.2.

Tabel 6.1. Kliimamooduli poolt edastatud andmed tavaolekus

Jrk. nr.	Bait 1	Bait 2	Bait 3	Bait 4	Bait 5	Bait 6	Bait 7	Bait 8
1	0x01	0x4B	0x00	0x12	0xE8	0x00	0x00	0x00
2	0x03	0x4B	0x00	0x12	0xE8	0x00	0x00	0x00

Tabel 6.2. Kliimamooduli poolt edastatud andmed peeglite klappimisnupu vajutades

Jrk. nr.	Bait 1	Bait 2	Bait 3	Bait 4	Bait 5	Bait 6	Bait 7	Bait 8
1	0x01	0x4F	0x00	0x12	0xE8	0x00	0x00	0x00
2	0x03	0x4F	0x00	0x12	0xE8	0x00	0x00	0x00

Tabelis 6.1 on näha, et tavaolekus saadab kliimaplokk pidevalt samu andmeid, ainukeseks vaheks ongi esimene bait, mis muutub vaheldumisi 0x01 ja 0x03 vahel. Samuti kliimaplokil peeglite klappimisnuppu vajutades edastatakse vastavalt hetkele kas tabeli 6.2 esimene rida või teine rida. Tabeleid võrreldes näeb, et nupu vajutusel muutub teine bait 0x4B-lt 0x4F-ile.

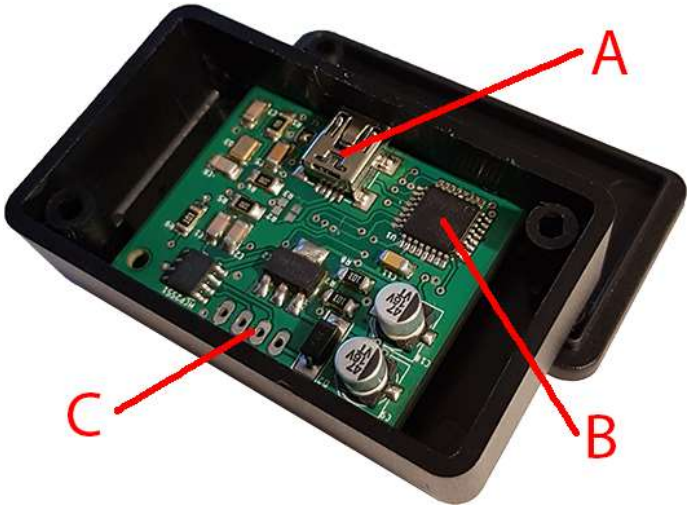
Teades, mida peab võrku edastama, et simuleerida küljepeeglite klappimis nupu vajutamist, saab lõpulikult koostada paketi. Paketi edastamisel esitleb seade ennast võrgule kui kliimaploki moodul. Selle jaoks määratakse edastava paketti ID-ks kliimaploki ID (milleks on 0x217FFC) ning edastatakse vastavalt hetkele, kas tabelis 6.2. oleva esimene või teine rida. Rea valik oleneb asjaolust, mis eelmisena moodul välja on saatnud. Kõik õigesti tehtult peaks nüüd peale paketi võrku sisestamist küljepeeglid ennast kokku klappima. Antud töö käigus üritati võimalikult palju esineda, kui diagnostika tarkvara, et mitte häirida moodulite tava tööolukorda. Eesmärk oli tõsta edastatava käskluse töökindlust, et pakett jõuaks soovitud kohta kohale, ning seade töötaks nii, kuidas see on programmeeritud.

Sama tehnoloogiat kasutades saab CAN-võrku edastada ka muid käsklusi, nagu näiteks tagurdustulede põlema panek pimedal ajal koos lombivalgustusega või uste lukustamine.

7. Lõpp-produkt

7.1. Valminud seade

Trükkplaat on paigaldatud eraldi karpi, kust väljub vaid sõidukisse minevad juhtmed. Tulevikus, kui on koostatud ka arvutile eraldi programm, mis oskab seadmega suhelda, siis seadme käitumises muudatuste tegemiseks on vajalik karp avada ning trükkplaadile külge ühendada USB juhe. Karpi on kujutatud joonisel 7.1. Joonisel tähis A tähistab USB pesa, B tähistab loogika mikrokontrollerit ning C tähistab sõidukijuhtmete ühenduspesasid.



Joonis 7.1. Seadme karp; A – USB pistik seadme käitumise muutmiseks; B – loogika mikrokontroller; C – sõidukijuhtmete ühenduspesad.

Tellitud trükkplaadi ja karbi mõõtmed varieeruvad tegelikkuses, seega ei mahtunud trükkplaat enam karbi sisse ära. Seega tuli karpi, kuhu trükkplaat asetatakse, vähesel määral modifitseerida – karbi kahte seina tuli teha õhemaks. Autor kasutas selleks teravat vaibanuga. Karbi sein, kus pool on USB pistik, tuli võta umbes 0,5 mm jagu rohkem maha, kuna USB pistik ulatub natukene üle trükkplaadi ääre.

Seadme juhtkoodi väljatöötamise muudab ajamahukaks asjaolu, et koodi juppi või funktsiooni kontrollimiseks, tuleb igakord eraldi programmeerimise juhtmed ühendada ning

tihtipeale murduvad juhtmekiud jootekoha juures ära ning need tuleb uuesti paigaldada. Samuti tuleb igakord jälgida, et ei tekkiks lühiseolukorda.

Peale sõiduki juhtmete jootmist plaadile lisatakse juhtmele üks kaabliside karbi sisemisele poolele, et kaablit kogemata tõmmates ei avalduks trükkplaadile ja joodistele üleliigset jõudu, mis võib juhtmed trükkplaadi küljest lahti tõmmata. Lahendust on kujutatud joonisel 7.2.

Joonis 7.2. Karbisisene ühendus koos sõidukisse minema juhtmetega; A – Kaabliside juhtmete kinni hoidmiseks.

Jooniselt on näha, et kaabliside on paigaldatud juhtmetele selliselt, et karbi sisse on jäetud varu trükkplaadi eemaldamiseks ning USB kaabli ühendamiseks.

7.2. Juhtkood

Loogika mikrokontrolleri juhtkood on kirjutatud Arduino lihtsustatud programmeerimiskeelel. Tegemist on C ja C++ programmeerimiskeelte seguga, millel on juures veel kohandatud elemente [14]. Juhtkoodi kirjutamisel kasutati SeeedStudio *CAN BUS Shield*-i teeki, kuna see eemaldas keerulise ajastus-, järjestus- ning lugemisprobleemid. Loogika mikrokontrolleri juhtkood on lisatud lisasse B. Lisas olev juhtkoodilt on eemaldatud käskluste pakettide sisud konfidentsiaalsuse huvides.

Sõidukile lisati võimalused:

- puldist aknad lahti ja kinni lasta;
- uste avamisel või lukustamisel küljepeeglid kinni või lahti;
- uste lukustamisel või lahti tegemisel panna tööle lombivalgusti;
- koos lombivalgustiga pannakse põlema ka tagurdustuled;
- automaatne uste lukustus sõitma hakkamisel.

Lisatud võimalustest täpsemalt räägitud järgnevates peatükkides.

7.3. Küljepeeglite klappimine

Sõiduki uste lahti tegemisel või lukustamisel küljepeeglite lahti või kinni klappimise jaoks kuulati pidevalt pealt CAN-võrku. Kui moodul, mille ID oli 0x1E0162A, edastas võrku paketi, mille neljas bait algas B tähega ning viies bait algas 2-ga ning bait oli kahekohaline, siis käivitati funktsioon, mis saadab võrku peeglite kokku klappimise käskluse. Kui sama moodul edastas võrku paketi, mille neljas bait algas „A“ tähega ning viies bait algas 2-ga ning bait oli kahekohaline, siis edastati peeglite lahti klappimise käsklus.

Paketi, mis tuleb võrku edastada, teada saamiseks kasutati diagnostika kuulamise meetodit. Antud paketi saatmise korral võrku on tähtis, et saadetakse ka nii-öelda lõpetamise käsklus, vastasel korral jäävad mootorid tööle ning võivad läbi põleda või tekitada muid anomaaliaid sõiduki süsteemi töös.

7.4. Akende avamine puldist

Puldist akende kinni või lahti tegemiseks kuulati sama ID-koodi, nagu kuulati küljepeeglite lahti või kinni klappimiseks. Ainsaks erinevuseks on see, et viies bait peab algama 0-ga ning sama pakett peab olema esitatud võrku kolm korda etteantud ajavahemikul (mis oli seadistatud kolme sekundi peale). Igakord, kui tingimused olid käes, jäeti meelde, et puldil on nuppu vajutatud. Kui sõiduki omanik vajutab kolme sekundi jooksul uuesti samale nupule, suurendatakse meelde jäetud muutuja väärtust ühe võrra. Kui sõiduki omanik on kokku vajutanud kolm korda, edastatakse võrku akende lahti või kinni tegemise käsklus, vastavalt kas vajutati puldil lukustamis- või avamisnuppu. Võrku edastati taaskord diagnostika kuulamise meetodil leitud pakett.

7.5. Lombivalgusti käivitamine uste avamisega/lukustamisega

Lombivalgusti puhul kuulati taaskord sama mooduli väärtuste muutumist, mida tehti akende kinni panemiseks või lahti tegemiseks. Siinkohal lisati ka juhtkoodi juurde võimalusel lisada valgusanduri väärtus. See tähendab seda, et kui leitakse üles õige moodul, mis edastab valgusanduri väärtuse, siis on võimalik lihtsalt juurde lisada võimalus, et lombivalgusti läheb põlema vaid juhul, kui väljas on alla teatud taseme valgust. Tulemuse saamiseks edastati võrku taaskord pakett, mis oli kuulatud diagnostika tarkvara ja arvuti vahelisest suhtlusest. Ka siin tuli saata võrku lõpetus käsklus, mis saadetakse 30 sekundi pärast.

Pimedas sõiduki tagumisest otsast lähenedes parema nähtavuse saamiseks lisati lisaks lombivalgustite käima minekule veel ka tagurdustuled. Tagurdustuled põlevad sama kaua, kui lombivalgustid ning ka see käsklus on kuulatud diagnostika meetodil. Tagurdustuled kustutati võrku saadetud lõpetus käsklusega.

7.6. Automaatne lukustus sõidu alustamisel

Automaatne lukustus sõitma hakkamisel lahendati selliselt, et kuulati CAN-võrku pidevalt pealt. Täpsemalt huvitab moodul, mille ID on 0x3200408, kuna see edastab perioodiliselt (umbes iga 90-100 millisekundi järelt) võrku andmeid, muuhulgas ka käiguvaheti positsioonist. Käiguvaheti bait oli seitsmes, mis muutus 0x10 ja 0x40 vahel (kus 0x10 tähistas P ehk *Parking*'ut, 0x20 tähistas R ehk *Reverse*, 0x30 tähistas N ehk *Neutral* ja 0x40 tähistas D ehk *Drive*'i). Jälgiti baiti, mis näitab, millises asendis on käiguvaheti ning niipea, kui väärtus muutus 0x10 või 0x40 peale, edastati võrku lukusta käsklus. Lukusta käsklus saadi CAN-võrgu kuulamise meetodil ning edastati seadme poolt võrku teeseldes, et on moodul, mille ID on 0x1601422. Peale käskluse edastamist oodati 300 millisekundit ning seejärel kontrolliti ukseluku asendit. Ukseluku asendi kontrollimiseks kuulati sama mooduli, millega edastati lukusta käsklus, kuuendat baiti. Kui bait lõppes numberiga 3, näitas see seda, et uksed on lukus, vastasel juhul näitas, et uksed on lukust lahti. Kui peale esmakordset lukusta käskluse saatmist võrku oli ukseluku asend endine ehk käskluse saatmine polnud edukas, saadeti võrku sama käsklus uuesti. Antud võimalus on mõeldud kuni 2005 aasta sõidukitele, uuematel on antud lisa võimalik aktiveerida nupukombinatsiooniga salongist.

KOKKUVÕTE

Kõige enam kasutati töö edukaks läbiviimiseks loogika analüsaatorit Saleae ning diagnostika tarkvara VIDA. Kokku tehti trükkplaadi kujundust neli erinevat versiooni, et saavutada võimalikult väike suurus. Lõpp kujunduse mõõtmeteks jäi 30x40 mm. Sõiduki suhtlust sai edukalt pealt kuulatud mõningate probleemidega. Nimelt ei olnud võimalik pealt kuulata osasid mooduleid, kuna need ei olnud ühendatud otseselt sinna võrku, kus oli mugavusmooduli seade. Soovitud käskluste võrku edastamiseks teeseldi huvi pakkuvat moodulit või diagnostika tarkvara.

Teatud moodulitega polnud võimalik saada ühendust, kuna nende suhtlus ei pääsenud keskelektroonika moodulist (CEM) mööda REMi poole. Sellega seoses ei olnud võimalik täita sissejuhatuses kirjutatud mõningaid soove. Näiteks ei olnud võimalik saada ühendust tulede lüliti mooduliga, päikesesensoriga, istmesoojenduste moodulitega ning sademeteandurile. Ilmselt aitaks siinkohal see, kui teeks seadme ühenduse OBDII ja keskelektroonika mooduli vahele, et pealt kuulata, mis käsklusi diagnostika tarkvara väljastab. Põhjus, miks suhtlus ei pääsenud edasi REMi poole on see, et antud mooduleid juhibki CEM ise.

Visuaalselt korrektse tulemuse saamiseks tuleks sõiduki ühendusjuhtmed asendada kaabliga. Samuti otsida sisemistelt mõõtmetelt natukene suurem karp, et ei peaks hakkama vana modifitseerima.

Uste lukustus funktsiooni võiks modifitseerida selliselt, et uksed lukustatakse alles kiirust arendades ning avatakse automaatselt, kui sõiduki seisab ning kui lükatakse käiguvaheti asendisse P. Hetkel ei olnud võimalik nii lahendada, kuna ei saadud kätte sõiduki kiiruse andmeid.

Samuti võiks lombivalgustid käivituda ainult juhul, kui väljas on tõesti pime, säästmaks üleliigset voolutarbimist ning pirnide eluiga. Samuti võiks tagumised tagurdustuled

rakenduda alles mingist valgustasemest. Lisana võiks juurde veel lisada, et koos lombivalgustitega läheks põlema ka eesmised udutuled.

Leides üles mooduli, mis edastab sademeteanduri infot, saaks ka juurde lisada funktsiooni, kus näiteks iga 5 minuti tagant kontrollitakse sademeid ning nende olemasolul suletakse automaatselt sõiduki aknad.

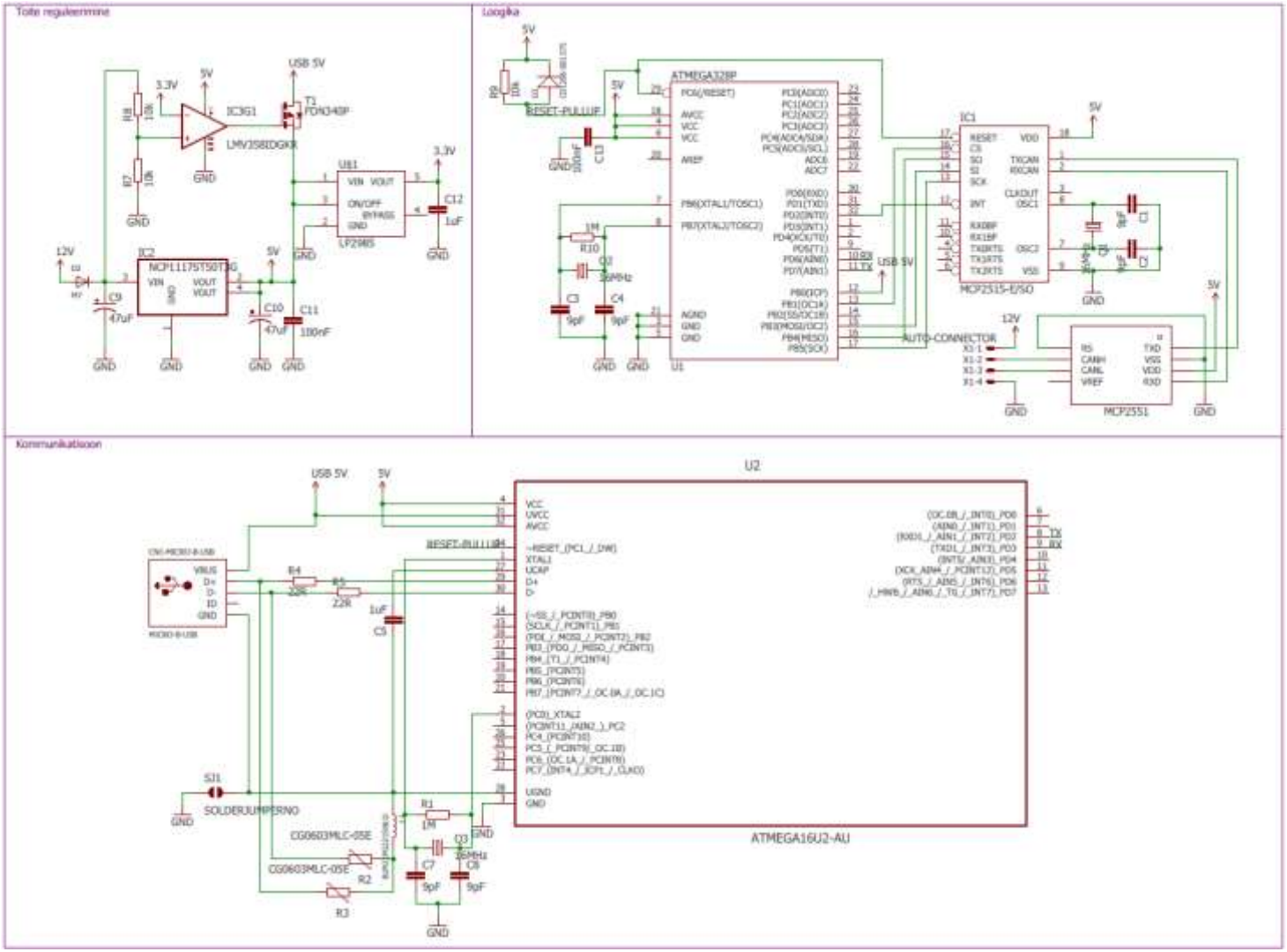
Veel võiks lisada, et alla 5 °C väli temperatuuri korral pannakse sõiduki käivitamisel käima lisaks istmesoojendusele veel ka tagumine klaasisoojendus. Mõlemat funktsiooni ei olnud võimalik hetkel töösse lisada, kuna ei saadud kätte sobivate moodulite ID-koode.

KASUTATUD KIRJANDUS

1. FCP Euro. Volvo P1, P2, and P3 - What's it All Mean? [veebileht] https://blog.fcpeuro.com/volvo-p1-p2-and-p3-whats-it-all-mean (13.05.2018).
2. Sewell Direct. A Brief Explanation of CAN Bus. [veebileht] https://sewelldirect.com/learning-center/canbus-technology (13.05.2018).
3. CSS Electronics. CAN BUS EXPLAINED - A SIMPLE INTRO (2018). [veebileht] https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en (14.05.2018).
4. **Murphy, N.** (2003). A short trip on the CAN bus. [artikkel] https://www.embedded.com/electronics-blogs/murphy-s-law/4024614/A-short-trip-on-the-CAN-bus (15.05.2018).
5. Orion BMS. Diagnosing CANBUS Communication Problems. [veebileht] https://www.orionbms.com/general/diagnosing-canbus-communication-problems/ (14.05.2018).
6. **Fischl, T.** USBasp - USB programmer for Atmel AVR controllers http://www.fischl.de/usbasp/ (15.05.2018).
7. **Camera, D.** LUFA (2013). http://www.fourwalledcubicle.com/LUFA.php (21.05.2018).
8. Arm Keil. USB Component - CDC: Communication Device Class. [veebileht] https://www.keil.com/pack/doc/mw/USB/html/_c_d_c.html (21.05.2018)
9. Seeed Studio. PCB. [veebileht] https://www.seeedstudio.com/fusion_pcb.html (13.05.2018)
10. Volvo diagnostikatarkvara VIDA
11. Digital Trends. From dongles to diagnostics, here’s all you need to know about OBD/OBD II. [veebileht] https://www.digitaltrends.com/cars/everything-you-need-to-know-about-obd-obdii/ (20.05.2018).
12. AVRdude. AVRDUDE - AVR Downloader/UploaDEr. [veebileht]. https://www.nongnu.org/avrdude/ (15.05.2018).
13. Engbedded. Engbedded Atmel AVR® Fuse Calculator. [veebileht] http://www.engbedded.com/fusecalc/ (14.05.2018).
14. **Gendel, I.** The Arduino programming language: Which one is it? (2014). http://www.idogendel.com/en/archives/19 (18.05.2018).

LISAD

Lisa A. Elektriskeemi üldvaade



Lisa B. Loogika mikrokontrolleri juhtkood

```
#include <SPI.h>
#include "mcp_can.h"

// Valikuvõimalused
bool LukustuselPeeglid = 1;
bool LukustuselValgustus = 1;
bool ValgusustelTagurdustuled = 1;
bool PuldiltAknadKinni = 1;
bool PuldiltAknadLahti = 1;
bool UsteLukustus = 1;
// Valikuvõimalused lõpp

int PeeglidAktiivneMillis = 3500;
int ValgustusAktiivneMillis = 30000;
int AknadAktiivneMillis = 4500;

byte VFold1[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte VFold2[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte VFoldStop[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

byte PFold1[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte PFold2[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte PFoldStop[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

byte VUnFold1[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte VUnFold2[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte VUnFoldStop[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

byte PUnFold1[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte PUnFold2[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte PUnFoldStop[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

byte VAkenAlla1[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte VAkenAlla2[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte VAkenAllaStop[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

byte PAkenAlla1[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte PAkenAlla2[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte PAkenAllaStop[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

byte VAkenUles1[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte VAkenUles2[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte VAkenUlesStop[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

byte PAkenUles1[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte PAkenUles2[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte PAkenUlesStop[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

byte ApproachValgus1[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}; // | Variant 1
byte ApproachValgus2[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}; // | Variant 1
byte ApproachValgusStop[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}; // | Variant 1

byte TagurdusTuli[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

Lisa B järg

```
byte TagurdusTuliStop[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

byte LockCommand[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
long unsigned int IDUpperMoodul = 0x1E0162A;
long unsigned int IDGearState = 0x3200408;
long unsigned int IDLock = 0x1601422;

bool PeeglidSaatmiseStaatus[5] = {0, 1, 1, 1, 1}; // 1-Suund (0-kinni, 1-lahti), 2,3,4,5-käskluse olek
bool ValgustusSaatmiseStaatus[2] = {1, 1};
bool AknadSaatmiseStaatus[5] = {0, 1, 1, 1, 1};
bool TagurdusSaatmiseStaatus[1] = {1};
bool LockSaatmiseStaatus[1] = {1};

bool PeeglidStopStaatus[2] = {0, 0};
bool ValgustusStopStaatus[1] = {0};
bool AknadStopStaatus[2] = {0, 0};
bool TagurdusStopStaatus[1] = {0};

bool StopPeeglid = false;
bool StopValgustus = false;
bool StopAknad = false;
bool StopTagurdus = false;

byte ViimasenaVajutatudPuldiSuund = 0; //1 - Kinni; 2- Lahti
byte KorvalistujaLukuAsend;

bool UksedSuletud = false;

unsigned long PeegliteAlgusMillis = 0;
unsigned long ValgustusAlgusMillis = 0;
unsigned long AknadAlgusMillis = 0;
unsigned long TagurdusAlgusMillis = 0;

int PuldiNupudVajutatud = 0;
int PuldiNupudTimeoutMillis = 3000;

unsigned long PuldiNupudLastMillis = 0;

bool kuulaCAN = true;

const int SPI_CS_PIN = 10; // Väljund, kus on MCP2515 CS jalg

MCP_CAN CAN(SPI_CS_PIN); // Seadistame SPI

void setup(){
  while (CAN_OK != CAN.begin(CAN_125KBPS)) // Alustame CAN-võrgu kuulamist kiiruselt 125 kbps
  {
    delay(100);
  }
}

void loop(){

  if(CAN_MSGAVAIL == CAN.checkReceive() && kuulaCAN){ // Kontrollime, kas on uusi andmeid
```

Lisa B järg

```
      unsigned char pikkus = 0;
      unsigned char puhver[8];
      unsigned long lugemisAeg = millis();
      CAN.readMsgBuf(&pikkus, puhver); // Loeme CAN-võrgust infot

      long unsigned int canID = CAN.getCanId();

      kontrolliData(canID, puhver);
   }

   PuldiNupudKontroll();
   SaadaStopCommand();
}

void kontrolliData(long unsigned int ID, byte buffer[8]){
   // Kontrollime, kas on uksed puldist avatud või lukustatud
   if(ID == IDUpperMoodul){
      if(KontrolliByte(buffer[3], "A", 1)){ // Uksed tehti puldist lahti
         ViimasenaVajutatudPuldiSuund = 2;

         if(KontrolliByte(buffer[4], "2", 1)){

            if(LukustuselPeeglid && PeeglidSaatmiseStaatus[0] != 1){ // Kui peeglite klappimine aktiveeritud
siis minna SaadaUnfold funktsiooni

               PeeglidSaatmiseStaatus[0] = 1;
               PeeglidSaatmiseStaatus[1] = 0;
               PeeglidSaatmiseStaatus[2] = 0;
               PeeglidSaatmiseStaatus[3] = 0;
               PeeglidSaatmiseStaatus[4] = 0;
               SaadaUnfold();
            }

            if(LukustuselValgustus && kasValjasPime()){
               ValgustusSaatmiseStaatus[0] = 0;
               ValgustusSaatmiseStaatus[1] = 0;
               LombiValgusti(); // Kui valgustus aktiveeritud ja õues pime siis minna LombiValgusti
funktsiooni
            }

         }
         else{
            PuldiNupudVajutatud++;
            PuldiNupudLastMillis = millis();
         }
      }
      else if(KontrolliByte(buffer[3], "B", 1)){ // Uksed pandi puldist kinni
         ViimasenaVajutatudPuldiSuund = 1;
         if(KontrolliByte(buffer[4], "2", 1)){

            if(LukustuselPeeglid && PeeglidSaatmiseStaatus[0] != 0){ // Kui peeglite klappimine aktiveeritud
siis minna SaadaFold funktsiooni

               PeeglidSaatmiseStaatus[0] = 0;
               PeeglidSaatmiseStaatus[1] = 0;
```

Lisa B järg

```
                PeeglidSaatmiseStaatus[2] = 0;
                PeeglidSaatmiseStaatus[3] = 0;

                PeeglidSaatmiseStaatus[4] = 0;
                SaadaFold();
            }

            if(LukustuselValgustus && kasValjasPime()){
                ValgustusSaatmiseStaatus[0] = 0;
                ValgustusSaatmiseStaatus[1] = 0;
                LombiValgusti(); // Kui valgustus aktiveeritud ja õues pime siis minna LombiValgusti
funktsiooni
            }

        }
        else{
            PuldiNupudVajutatud++;
            PuldiNupudLastMillis = millis();
        }
    }
  }
  if(ID == IDGearState){
    if(KontrolliByte(buffer[6], "1", 1)){
      // Käiguvaheti on P asendis

      if(UksedSuletud){ // Kontrollime, kas uksed on veel suletud
        // Teeme uksed lukust lahti
        LockSaatmiseStaatus[0] = 0;
        SaadaLock();
        delay(300);

        //Kontrollime, kas uksed said lahti
        if(KontrolliLukuAsendit()){
          //Uksed on lukus, teeme uuesti lahti
          LockSaatmiseStaatus[0] = 0;
          SaadaLock();

        }

        UksedSuletud = 0;
      }
    }
    else if(KontrolliByte(buffer[6], "4", 1)){
      // Käiguvaheti on D asendis

      if(!UksedSuletud){
        // Paneme uksed lukku
        LockSaatmiseStaatus[0] = 0;
        SaadaLock();
        delay(300);

        //Kontrollime, kas uksed lukus
        if(!KontrolliLukuAsendit()){
          //Uksed ei ole lukus, lukustame uuesti
          LockSaatmiseStaatus[0] = 0;
```

Lisa B järg

```
                SaadaLock();

            }

            UksedSuletud = 1;
        }
    }
  }
  if(ID == IDLock){
    KorvalistujaLukuAsend = buffer[5];
  }

}

bool KontrolliByte(byte KontrollitavByte, String Vaartus, int Indeks){ // Kontrollime, kas baidil, soovitud kohas, on väärtus, mis edastati funktsiooniga

  // Muudame baidi sõneks, et saaks lihtsamalt modifitseerida seda
  String ajutineString = String(KontrollitavByte, HEX);

  // Kontrollime, kas sõne sisaldab endas kahte karakterit, kui ei, siis lisame
  if(ajutineString.length() == 1){
    ajutineString = "0" + ajutineString;
  }
  //Kas soovitakse kontrollida esimest(vasakpoolset) väärtust
  if(Indeks == 1){
    ajutineString = ajutineString.substring(0, 1);
  }
  else{
    ajutineString = ajutineString.substring(1);
  }
  //Muudame sõne suureks täheks
  ajutineString.toUpperCase();
  if(ajutineString == Vaartus){
    return(1); // Kontroll tõene
  }
  else{
    return(0); // Kontroll väär
  }
}

void SaadaUnfold(){

  // Vasak peegel
  if(PeeglidSaatmiseStaatus[1] != 1){ // Kontrollime, kas antud käsklus edastati edukalt
    byte peeglidSaatmiseStaatus1 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, VUnFold1);
    delay(10);

    if(peeglidSaatmiseStaatus1 == CAN_OK) PeeglidSaatmiseStaatus[1] = 1;
  }
  if(PeeglidSaatmiseStaatus[2] != 1){ // Kontrollime, kas antud käsklus edastati edukalt
    byte peeglidSaatmiseStaatus2 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, VUnFold2);
    delay(10);

    if(peeglidSaatmiseStaatus2 == CAN_OK) PeeglidSaatmiseStaatus[2] = 1;
```

Lisa B järg

```
    }
    // Parem peegel
    if(PeeglidSaatmiseStaatus[3] != 1){ // Kontrollime, kas antud käsklus edastati edukalt

        byte peeglidSaatmiseStaatus3 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, PUnFold1);
        delay(10);

        if(peeglidSaatmiseStaatus3 == CAN_OK) PeeglidSaatmiseStaatus[3] = 1;
    }
    if(PeeglidSaatmiseStaatus[4] != 1){ // Kontrollime, kas antud käsklus edastati edukalt
        byte peeglidSaatmiseStaatus4 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, PUnFold2);
        delay(10);

        if(peeglidSaatmiseStaatus4 == CAN_OK) PeeglidSaatmiseStaatus[4] = 1;
    }

    if(PeeglidSaatmiseStaatus[1] || PeeglidSaatmiseStaatus[2] || PeeglidSaatmiseStaatus[3] || PeeglidSaatmiseS
taatus[4]){
        StopPeeglid = true;
        PeeglidStopStaatus[0] = 0;
        PeeglidStopStaatus[1] = 0;
        PeegliteAlgusMillis = millis();
    }
}

void SaadaFold(){

    // Vasak peegel
    if(PeeglidSaatmiseStaatus[1] != 1){
        byte peeglidSaatmiseStaatus1 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, VFold1);
        delay(10);

        if(peeglidSaatmiseStaatus1 == CAN_OK) PeeglidSaatmiseStaatus[1] = 1;
    }
    if(PeeglidSaatmiseStaatus[2] != 1){
        byte peeglidSaatmiseStaatus2 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, VFold2);
        delay(10);

        if(peeglidSaatmiseStaatus2 == CAN_OK) PeeglidSaatmiseStaatus[2] = 1;
    }

    // Parem peegel
    if(PeeglidSaatmiseStaatus[3] != 1){
        byte peeglidSaatmiseStaatus3 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, PFold1);
        delay(10);

        if(peeglidSaatmiseStaatus3 == CAN_OK) PeeglidSaatmiseStaatus[3] = 1;
    }
    if(PeeglidSaatmiseStaatus[4] != 1){
        byte peeglidSaatmiseStaatus4 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, PFold2);
        delay(10);

        if(peeglidSaatmiseStaatus4 == CAN_OK) PeeglidSaatmiseStaatus[4] = 1;
    }
```

Lisa B järg

```
   if(PeeglidSaatmiseStaatus[1] || PeeglidSaatmiseStaatus[2] || PeeglidSaatmiseStaatus[3] || PeeglidSaatmiseS
taatus[4]){
      StopPeeglid = true;
      PeeglidStopStaatus[0] = 0;

      PeeglidStopStaatus[1] = 0;
      PeegliteAlgusMillis = millis();
   }
}

void LombiValgusti(){

   if(ValgustusSaatmiseStaatus[0] != 1){
      byte valgustusSaatmiseStaatus1 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, ApproachValgus1);
      delay(10);

      if(valgustusSaatmiseStaatus1 == CAN_OK) ValgustusSaatmiseStaatus[0] = 1;
   }
   if(ValgustusSaatmiseStaatus[1] != 1){
      byte valgustusSaatmiseStaatus2 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, ApproachValgus2);
      delay(10);

      if(valgustusSaatmiseStaatus2 == CAN_OK) ValgustusSaatmiseStaatus[1] = 1;
   }
   if(ValgusustelTagurdustuled){
      TagurdusSaatmiseStaatus[0] = 0;
      TagurdusTuled();
   }

   if(ValgustusSaatmiseStaatus[0] && ValgustusSaatmiseStaatus[1]){
      StopValgustus = true;
      ValgustusStopStaatus[0] = 0;
      ValgustusStopStaatus[1] = 0;
      ValgustusAlgusMillis = millis();
   }
}

void TagurdusTuled(){

   if(TagurdusSaatmiseStaatus[0] != 1){
      byte tagurdusSaatmiseStaatus1 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, TagurdusTuli);
      delay(10);

      if(tagurdusSaatmiseStaatus1 == CAN_OK) TagurdusSaatmiseStaatus[0] = 1;
   }
   if(TagurdusSaatmiseStaatus[0]){
      StopTagurdus = true;
      TagurdusStopStaatus[0] = 0;
      TagurdusAlgusMillis = millis();

   }
}

void SaadaLock(){
```

Lisa B järg

```
    if(LockSaatmiseStaatus[0] != 1){
      byte SaadaLockStaatus1 = CAN.sendMsgBuf(IDLock, 1, 8, LockCommand);
      delay(10);

      if(SaadaLockStaatus1 == CAN_OK) LockSaatmiseStaatus[0] = 1;

    }
}

bool KontrolliLukuAsendit(){
    return(KontrolliByte(KorvalistujaLukuAsend, "3", 2));
}

bool kasValjasPime(){
    // Siia valgusanduri väärtusest tulenevalt tagastab kas 1(tõene) või 0(väär)
    // Hetkel pole leitud õiget moodulit, mis seda väljastaks
    return(1);
}

void PuldiNupudKontroll(){
    // Serial.println(PuldiNupudVajutatud);
    if(PuldiNupudVajutatud != 0){
      if(abs(millis() - PuldiNupudLastMillis) > PuldiNupudTimeoutMillis){
        PuldiNupudVajutatud = 0;
      }
    }
    if(PuldiNupudVajutatud >= 3){
      //Avame/sulegme aknad

      if(ViimasenaVajutatudPuldiSuund == 1){
        //Sulgeme aknad

        if(PuldiltAknadKinni){
          AknadSaatmiseStaatus[1] = 0;
          AknadSaatmiseStaatus[2] = 0;
          AknadSaatmiseStaatus[3] = 0;
          AknadSaatmiseStaatus[4] = 0;
          Aknad(0);
        }

        PuldiNupudVajutatud = 0;
      }
      else if(ViimasenaVajutatudPuldiSuund == 2){
        //Avame aknad

        if(PuldiltAknadLahti){
          AknadSaatmiseStaatus[1] = 0;
          AknadSaatmiseStaatus[2] = 0;
          AknadSaatmiseStaatus[3] = 0;
          AknadSaatmiseStaatus[4] = 0;
          Aknad(1);
        }

        PuldiNupudVajutatud = 0;
      }
```

Lisa B järg

```
    }
}

void Aknad(bool Suund){
  if(Suund){ // Avame aknad
    AknadSaatmiseStaatus[0] = 1;

    if(AknadSaatmiseStaatus[1] != 1){
      byte saatmiseStaatus1 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, VAkenAlla1);
      delay(10);

      if(saatmiseStaatus1 == CAN_OK) AknadSaatmiseStaatus[1] = 1;
    }
    if(AknadSaatmiseStaatus[2] != 1){
      byte saatmiseStaatus2 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, VAkenAlla2);
      delay(10);

      if(saatmiseStaatus2 == CAN_OK) AknadSaatmiseStaatus[2] = 1;
    }
    if(AknadSaatmiseStaatus[3] != 1){
      byte saatmiseStaatus3 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, PAkenAlla1);
      delay(10);

      if(saatmiseStaatus3 == CAN_OK) AknadSaatmiseStaatus[3] = 1;
    }
    if(AknadSaatmiseStaatus[4] != 1){
      byte saatmiseStaatus4 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, PAkenAlla2);
      delay(10);

      if(saatmiseStaatus4 == CAN_OK) AknadSaatmiseStaatus[4] = 1;
    }

    if(AknadSaatmiseStaatus[1] || AknadSaatmiseStaatus[2] || AknadSaatmiseStaatus[3] || AknadSaatmiseSt
aatus[4]){
      StopAknad = true;
      AknadStopStaatus[0] = 0;
      AknadStopStaatus[1] = 0;
      AknadAlgusMillis = millis();
    }

  }
  else{ // Sulgeme aknad

    AknadSaatmiseStaatus[0] = 0;

    if(AknadSaatmiseStaatus[1] != 1){
      byte aknadSaatmiseStaatus1 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, VAkenUles1);
      delay(10);

      if(aknadSaatmiseStaatus1 == CAN_OK) AknadSaatmiseStaatus[1] = 1;
    }
    if(AknadSaatmiseStaatus[2] != 1){
      byte aknadSaatmiseStaatus2 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, VAkenUles2);
      delay(10);
```

Lisa B järg

```
        if(aknadSaatmiseStaatus2 == CAN_OK) AknadSaatmiseStaatus[2] = 1;
      }
      if(AknadSaatmiseStaatus[3] != 1){
        byte aknadSaatmiseStaatus3 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, PAkenUles1);
        delay(10);

        if(aknadSaatmiseStaatus3 == CAN_OK) AknadSaatmiseStaatus[3] = 1;

      }
      if(AknadSaatmiseStaatus[4] != 1){
        byte aknadSaatmiseStaatus4 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, PAkenUles2);
        delay(10);

        if(aknadSaatmiseStaatus4 == CAN_OK) AknadSaatmiseStaatus[4] = 1;
      }

      if(AknadSaatmiseStaatus[1] || AknadSaatmiseStaatus[2] || AknadSaatmiseStaatus[3] || AknadSaatmiseSt
aatus[4]){
        StopAknad = true;
        AknadStopStaatus[0] = 0;
        AknadStopStaatus[1] = 0;
        AknadAlgusMillis = millis();
      }

    }
}

void SaadaStopCommand(){
  if(StopPeeglid){
    if(abs(millis()-PeegliteAlgusMillis) >= PeeglidAktiivneMillis){

      if(PeeglidStopStaatus[0] != 1){
        byte stopSaatmiseStaatus1 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, VFoldStop);
        delay(10);

        if(stopSaatmiseStaatus1 == CAN_OK){
          PeeglidStopStaatus[0] = 1;
        }
      }
      if(PeeglidStopStaatus[1] != 1){
        byte stopSaatmiseStaatus2 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, PFoldStop);
        delay(10);

        if(stopSaatmiseStaatus2 == CAN_OK){
          PeeglidStopStaatus[1] = 1;
        }
      }
      if(PeeglidStopStaatus[0] && PeeglidStopStaatus[1]){
        StopPeeglid = 0;
      }
    }
  }

  if(StopValgustus){
    if(abs(millis()-ValgustusAlgusMillis) >= ValgustusAktiivneMillis){
```

Lisa B järg

```
        if(ValgustusStopStaatus[0] != 1){

          byte stopSaatmiseStaatus1 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, ApproachValgusStop);
          delay(10);

          if(stopSaatmiseStaatus1 == CAN_OK){
            ValgustusStopStaatus[0] = 1;

          }
        }

        if(ValgustusStopStaatus[0]){
          StopValgustus = 0;
        }
      }
    }

    if(StopAknad){
      if(abs(millis() - AknadAlgusMillis) >= AknadAktiivneMillis){

        if(AknadStopStaatus[0] != 1){
          byte stopSaatmiseStaatus1 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, VAkenAllaStop);
          delay(10);

          if(stopSaatmiseStaatus1 == CAN_OK){
            AknadStopStaatus[0] = 1;
          }
        }
        if(AknadStopStaatus[1] != 1){
          byte stopSaatmiseStaatus2 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, PAkenAllaStop);
          delay(10);

          if(stopSaatmiseStaatus2 == CAN_OK){
            AknadStopStaatus[1] = 1;

          }
        }
        if(AknadStopStaatus[0] && AknadStopStaatus[1]){
          StopAknad = 0;

        }
      }
    }

    if(StopTagurdus){
      if(abs(millis()-ValgustusAlgusMillis) >= ValgustusAktiivneMillis){

        if(TagurdusStopStaatus[0] != 1){

          byte stopSaatmiseStaatus1 = CAN.sendMsgBuf(0x000FFFFE, 1, 8, TagurdusTuliStop);
          delay(10);

          if(stopSaatmiseStaatus1 == CAN_OK){
            TagurdusStopStaatus[0] = 1;
```

Lisa B järg

```
            }
        }

        if(TagurdusStopStaatus[0]){
            StopTagurdus = 0;

        }
    }
  }
}
```

Lihtlitsents lõputöö salvestamiseks ja üldsusele kättesaadavaks tegemiseks ning juhendaja(te) kinnitus lõputöö kaitsmisele lubamise kohta

Mina, _____,
(*autori nimi*)
sünniaeg _____,

1. annan Eesti Maaülikoolile tasuta loa (lihtlitsentsi) enda koostatud lõputöö

_____
_____
_____,
(*lõputöö pealkiri*)

mille juhendaja(d) on_____,
(*juhendaja(te) nimi*)

1.1. salvestamiseks säilitamise eesmärgil,
1.2. digiarhiivi DSpace lisamiseks ja
1.3. veebikeskkonnas üldsusele kättesaadavaks tegemiseks

kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile;
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Lõputöö autor _____
(*allkiri*)

Tartu, _____
(*kuupäev*)

Juhendaja(te) kinnitus lõputöö kaitsmisele lubamise kohta

Luban lõputöö kaitsmisele.

_____ _____
(*juhendaja nimi ja allkiri*) (*kuupäev*)

_____ _____
(*juhendaja nimi ja allkiri*) (*kuupäev*)